(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
22 February 2001 (22.02.2001)

PCT

(10) International Publication Number
WO 01/13257 A1

(51) International Patent Classification[7]: G06F 15/16

(21) International Application Number: PCT/AU00/00983

(22) International Filing Date: 17 August 2000 (17.08.2000)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:

PQ 2259	17 August 1999 (17.08.1999)	AU
PQ 2442	25 August 1999 (25.08.1999)	AU
PQ 2565	1 September 1999 (01.09.1999)	AU
PQ 3620	22 October 1999 (22.10.1999)	AU
09/474,468	29 December 1999 (29.12.1999)	US

(71) Applicant *(for all designated States except US)*: **BULLANT TECHNOLOGY PTY. LTD.** [AU/AU]; Level 5, 181 Miller Street, North Sydney, NSW 2059 (AU).

(72) Inventor; and

(75) Inventor/Applicant *(for US only)*: **CAHILL, Michael, James** [AU/AU]; 14 Regent Street, Summer Hill, NSW 2131 (AU).

(74) Agent: **WALLINGTON-DUMMER**; P.O. Box 297, Rydalmere, NSW 1701 (AU).

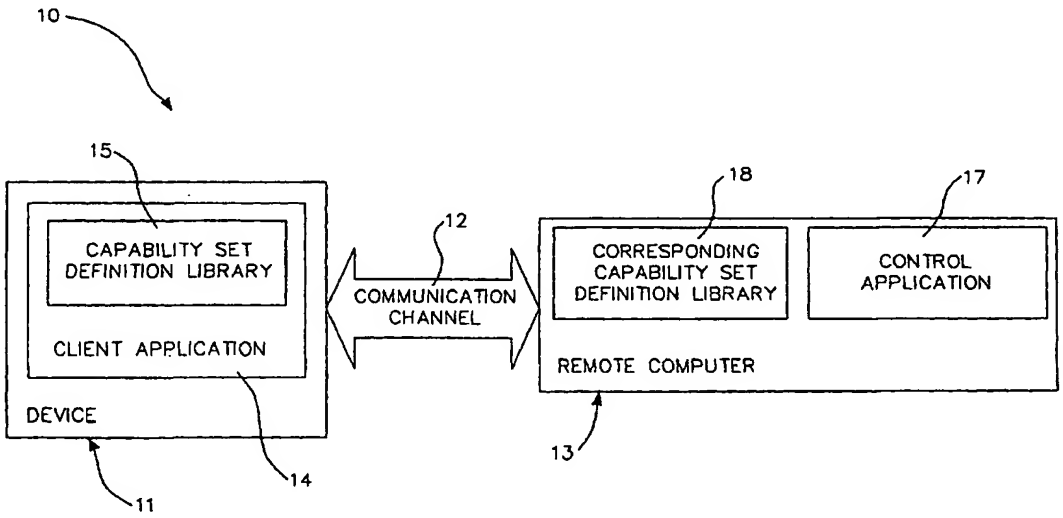
(81) Designated States *(national)*: AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, BZ, CA, CH, CN, CR, CU, CZ, DE, DK, DM, DZ, EE, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, MZ, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TR, TT, TZ, UA, UG, US, UZ, VN, YU, ZA, ZW.

(84) Designated States *(regional)*: ARIPO patent (GH, GM, KE, LS, MW, MZ, SD, SL, SZ, TZ, UG, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GW, ML, MR, NE, SN, TD, TG).

Published:
— *With international search report.*
— *With amended claims.*

For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.

(54) Title: A DISTRIBUTED SYSTEM FOR COMPUTER INTERACTION



(57) Abstract: A distributed system for computer interaction; the system including: a device having computing means therein and a remote computer located or locatable remotely from the device; communication means whereby bi-directional communication is established between the device and the remote computer; the device arranged to execute a client application; the remote computer arranged to execute a control application; and wherein communication between the device and the remote computer is assisted by use of a capability set definition library or model related to the client application.

WO 01/13257 A1

A DISTRIBUTED SYSTEM FOR COMPUTER INTERACTION

The present invention relates to a distributed system for computer interaction and, more particularly, to such a system and components therefor, for example when operating in a client/server model which provides increased capability for the client whilst also providing control over the volume of client/server data interchange.

BACKGROUND

The client/server model for interaction between computers or computer based systems is well established. A problem with such arrangements is that functions executed by the client which rely on delivery of data from the server can give rise to the transfer of unexpectedly large volumes of data over communication networks between the server and the client and over which there is no control once the function has been put in train. Given the highly distributed nature of many networks these days where it is not uncommon for the server to be located, literally, on the other side of the world from the client it follows that this characteristic of known systems can give rise to highly variable response times for the system as a whole and may, in some circumstances, and unpredictably so, lead to complete failure of the interaction between client and server.

A separate problem although sometimes related to the data transfer problem referred to above, concerns a reduction in capability of the client when operating under a client/server environment as compared with the capability of the client when operating on its own.

While the above problems have been referenced within the context of a client/server model, such problems can be encountered wherever and whenever two or more distributed

computing systems need to interact.

It is an object of the present invention to provide a system and components therefor which seek to address or ameliorate one or more of these perceived problems.

BRIEF DESCRIPTION OF INVENTION

Accordingly, in accordance with one aspect of the invention there is provided a distributed system for computer interaction; said system including

a device having computing means therein and

a remote computer located or locatable remotely from said device,

said device arranged to execute a client application which makes reference to a client capability set definition library located on said device;

said remote computer arranged to execute a control application which makes reference to a corresponding capability set definition library located on said remote computer.

In yet a further broad form of the invention there is provided a distributed client/server system for at least a first device and

a first remote computer;

communication means whereby bi-directional communication is established between said device and said remote computer;

said first device arranged to execute a client application which makes reference to an at least first client capability set driver or model;

said at least first remote computer arranged to execute a control application which makes reference to an at least first capability set definition library or

corresponding model;

said system operating sequentially in the following steps:

(a) said client application on said device initiates communication with said control application on said remote computer for the purpose of establishing a communications session;

(b) said client application and said control application agree on a capability set definition library or model and a corresponding client capability set driver or corresponding model for use during said communication session;

(c) said control application and said client application maintaining communication during said communication session whereby predetermined aspects of operation of said device are determined by said control application with reference to said selected capability set definition library or model and said corresponding selected client capability set driver or corresponding model;

said system arranged to communicate codes over said communication means which make reference to capability definitions in said capability set definition library or model thereby to control the volume of traffic on said communication means.

In yet a further broad form of the invention there is provided a method and system for facilitating a computer controlling an interaction with a device using a model of the device:

- The said device asking for a connection to the said computer

- The said computer creating a complete model being an abstraction of the said device on the said computer
- The said computer determining and imposing the appropriate behaviour information of the said device model abstraction on the said computer
- The said computer changing the said device model abstraction on the said computer to reflect the said imposed appropriate behaviour information of the said device model abstraction on the said computer
- The said computer transmitting the said imposed appropriate behaviour information to the said device
- The said device presenting the said imposed appropriate behaviour information
- The said device transmitting appropriate response behaviour information to the said computer
- The said computer receiving the said appropriate response behaviour information from the said device and transmitting determined and imposed appropriate behaviour information as a response.

Preferably said system operates sequentially in the following steps:

(a) said client application on said device initiates communication with said control application on said remote computer for the purpose of establishing a communications session between said device and said remote computer;

(b) said client application and said control application agree on mutual use of said capability

set definition library;

(c) said control application and said client application maintaining communication during said communication session whereby predetermined aspects of operation of said device are determined by said control application with reference to said capability set definition library

In accordance with a further aspect of the invention there is provided a distributed client/server system for at least

a first device and

a first remote computer

said first device arranged to execute a client application which makes reference to an at least first client capability set driver;

said at least first remote computer arranged to execute a control application which makes reference to an at least first capability set definition library;

said system operating sequentially in the following steps:

(a) said client application on said device initiates communication with said control application on said remote computer for the purpose of establishing a communications session;

(b) said client application and said control application agree on a capability set definition library and a corresponding client capability set driver for use during said communication session;

(c) said control application and said client application maintaining communication during said communication session whereby predetermined aspects of operation of said device are determined by said control application with reference to said selected

capability set definition library and said corresponding selected client capability set driver.

Preferably said capability set definition library contains a set of definitions of predetermined, selected capabilities of said device.

Preferably said predetermined capabilities are primitive functions of the device.

Preferably said capabilities are selected in order to minimise traffic during said communications session. Preferably said communications session includes the communicating of update information pertinent to the status of individual ones of said capabilities invoked with the aid of said capability set definition library.

Preferably said communications session occurs over a network link.

In accordance with yet a further aspect of the present invention there is provided a distributed system for computer interaction; said system including

a device having computing means therein and

a remote computer located or locatable remotely from said device,

communication means whereby bi-directional communication is established between said device and said remote computer; said device arranged to execute a client application; said remote computer arranged to execute a control application.

Preferably said device makes reference to a client capability set definition library located on said device. Preferably said control application makes reference to said capability set definition library.

Preferably said capability set definition library resides on both said device and said remote computer.

Preferably said system operates sequentially in the following steps:

(a) said client application on said device initiates communication with said control application on said remote computer for the purpose of establishing a communications session between said device and said remote computer;

(b) said client application and said control application agree on mutual use of said capability set definition library;

(c) said control application and said client application maintaining communication during said communication session whereby predetermined aspects of operation of said device are determined by said control application with reference to said capability set definition library.

Preferably said capability set definition library contains a set of definitions of predetermined, selected capabilities of said device.

Preferably said predetermined capabilities are primitive functions of the device.

Preferably said capabilities are selected in order to minimise traffic during said communication session. Preferably said communication session includes the communicating of update information pertinent to the status of individual ones of the capabilities invoked with the aid of said capability set definition library.

In accordance with yet a further aspect of the invention there is provided in a distributed system for computer interaction; said system including

a device having computing means therein and

a remote computer located or locatable remotely from

said device,

said device arranged to execute a client application;

said remote computer arranged to execute a control application; a method of operating said system according to the following:

(a) making available a capability set definition library on both said device and said remote computer;

(b) said client application on said device initiates communication with said control application on said remote computer for the purpose of establishing a communications session between said device and said remote computer;

(c) said client application and said control application agree on mutual use of said capability set definition library;

(d) said control application and said client application maintaining communication during said communication session whereby predetermined aspects of operation of said device are determined by said control application with reference to said capability set definition library.

Preferably said remote computer builds a model of the state of said client application so as to interact with said device by interacting with said model.

Preferably said model is continuously updated so as to reflect current status of said client application on said device.

Preferably said device is arranged to execute said client application with reference to a client capability set definition library located on said device and said remote computer is arranged to execute said control application by

making reference to a corresponding capability set definition library located on said remote computer.

Preferably said capability set definition library comprises a plurality of capability definitions, each of said definitions determined with reference to the desired functionality of said device.

Preferably each of said capability definitions comprises a primitive function of said device.

In an alternative preferred form each of said capability definitions comprises a combination of primitive functions of said device.

In accordance with yet a further aspect of the invention there is provided a distributed client/server system for at least

a first device and

a first remote computer;

communication means whereby bi—directional communication is established between said device and said remote computer;

said first device arranged to execute a client application which makes reference to an at least first client capability set driver;

said at least first remote computer arranged to execute a control application which makes reference to an at least first capability set definition library;

said system operating sequentially in the following steps:

(a) said client application on said device initiates communication with said control application on said remote computer for the purpose of establishing a communications session;

(b) said client application and said control

application agree on a capability set definition library and a corresponding client capability set driver for use during said communication session;

(c) said control application and said client application maintaining communication during said communication session whereby predetermined aspects of operation of said device are determined by said control application with reference to said selected capability set definition library and said corresponding selected client capability set driver;

said system arranged to communicate codes over said communication means which make reference to capability definitions in said capability set definition library thereby to control the volume of traffic on said communication means.

In yet a further broad form of the invention there is provided a method of implementing a client user interface for an input/output device which is adapted to be interconnected with a software application, the method comprising the steps of:

defining a first series of classes that simulate a model of the functional capabilities of the input/output device;

defining a second series of classes corresponding to the programs running on said device;

wherein to perform an action, said software application interacts with said first series of classes and upon an interaction, said first series of classes sends output command events corresponding to said action to said second series of classes running on said device; and

said second series of classes sends input command events to said first series of classes which translates said input command events into corresponding input actions and notifies said software application of said input action.

Preferably said input/output device comprises one of a palm computer, a mobile phone, a handheld computer or a desktop computer.

In yet a further broad form of the invention there is provided a method of implementing an interaction between an input/output device and at least one software application running on a server computer, the method comprising the steps of:

interconnecting said input/output device with said software application;

said input/output device and said software application negotiating a set of input/output capabilities for utilization in interacting with said input/output device;

said software application constructing a series of objects corresponding to said set of input/output capabilities and interacting with said series of objects with the interaction resulting in the objects sending output command events corresponding to said interaction to said input/output device; and

said input/output device sending input command events to said series of objects which translates said input command events into corresponding input actions and notifies said software application of said input action.

Preferably said input/output device interacts with multiple software applications.

In yet a further broad form of the invention there is provided a system when implementing the method described above.

In yet a further broad form of the invention there is provided a device which includes a client capability set definition library; said device adapted to be in communication with a remote computer.

In yet a further broad form of the invention there is provided a remote computer arranged to execute a control application which makes reference to a capability set definition library or model.

In yet a further broad form of the invention there is provided media incorporating software which implements the system described above.

In yet a further broad form of the invention there is provided media incorporating software which implements the method described above.

BRIEF DESCRIPTION OF DRAWINGS

Embodiments of the invention will now be described with reference to the accompanying drawings wherein:

Fig. 1 is a block diagram of a flexible, distributed system for computer interaction according to a first embodiment of the invention;

Fig. 2 is a diagram of an example client device and an example server remote computer implementing a specific graphical function interchange under the system of Fig. 1;

Fig. 3 is an exemplary block of program code for a control application running on the server remote computer of Fig. 1 which implements the graphical function interchange illustrated in Fig. 2;

Fig. 4 is a timing, step diagram showing the steps

involved in a typical interchange between the client device and server remote computer of Fig. 1 to implement the example of Fig. 2;

Figs. 5A-5K illustrate in block diagram form more detailed steps of the interaction and data transfer between the server remote computer and client device of Fig. 1 which give effect to the example of Fig. 2.

Fig. 6 is a block diagram of a flexible, distributed client/server system according to a second embodiment of the invention;

Fig. 7 is a block diagram of a multiple client/multiple application scenario possible with the embodiment of either Fig. 1 or Fig. 6;

Fig. 8 is a block diagram of a capability set definition library according to a further embodiment of the invention;

Fig. 9 is a block diagram of a capability set definition library according to yet a further preferred embodiment of the invention;

Fig. 10 illustrates an exemplary implementation scenario of an embodiment of the present invention;

Fig. 11 is a block diagram of a system according to a sixth embodiment of the invention; and

Fig. 12 is a block diagram of a system according to a seventh embodiment of the invention.

DETAILED DESCRIPTION OF PREFERRED EMBODIMENTS

Preamble

With reference to the drawings, embodiments of a system 10, 110 incorporate a new way of thinking about how arbitrary devices or clients are incorporated into computer systems. This has led to the implementation of a system that allows, for example, an application on a distant or remote computer

13, 113 to create a graphical user interface (GUI) on a (remote) display 16, 116 device and to handle the interactions with a person using the device 11, 111 on which the display is located.

Distributed Systems

One of the most basic problems in the construction of large systems concerns communication networks. Systems that involve only a single computer are much easier to design and understand than systems that involve interactions between autonomous execution environments. An understanding of single computer design does not scale simply into an understanding of distributed system design.

When thinking about any computer system, there are some questions whose answers are often enlightening:

- Where is the data and how is it represented?
- Where is the application logic?
- How does the data change over time?
- How does the system interact with its external environment?

A system that has clear answers to these questions has a chance of working in a production environment. Unfortunately, few of the large systems built so far satisfy this condition. Instead:

- Data is stored in databases, separate from the "application" and in a form that must be transformed when it crosses the boundary between the database and the rest of the system.
- Pieces of application logic are scattered around the system, from database tables to application servers to clients.
- Many parts of the system cause changes to the data.

- User interactions are an afterthought or are grafted onto inappropriate interfaces.

DESIRED ATTRIBUTES OF PREFERRED EMBODIMENTS

It is perceived as desirable to be able to create a mechanism that will allow interaction with arbitrary devices connected to a remote computer via a network using as little bandwidth as possible. In particular, it is desirable to be able to allow interaction between control applications on the remote computer and a client such as:

- any desktop computer that can run a web browser;
- handheld computers (Palm Computing devices, Windows CE™); and
- mobile phones.

Desirably the interactions should work over low bandwidth connections. In particular, it is desirable to use the system over a 9600 baud modem line (the communication rate of current mobile phone networks)

Desirably, the mechanism should be "natural" for application programmers to use. They should not have to learn about the details of how the devices communicate with the system. They should be able to make use of the capabilities of external devices without losing the idea that they are building a direct model of an application. Details should not influence design.

So, to summarise, desirable attributes are:

- to support a wide range of devices, connected via a network;
- to use low bandwidth; and
- to provide a simple, familiar programming environment for application developers.

Various embodiments of the present invention achieve one or more of these desired attributes, broadly speaking, by the adoption of an approach whereby both the remote computer (server) and client application agree to adopt a "capability set definition library". The adoption of this library permits the communication between the computers of short codes which can trigger much higher level data manipulation and utilisation with reference to a specified member or members of the capability set definition library. The members of the capability set definition library can be chosen or formulated with a view to the volume of communication traffic comprising the codes which can be tolerated for any given application and the likely communications environment in which it is expected to operate. So, for example, a local intranet or network having a high band width and reliable communications system in place could be expected to tolerate a much higher volume of data comprising or relating to the codes than a system which may have to operate over low band width dial up modems or the like. The concept of a capability set definition library is defined more fully immediately below and by way of example provided by the various embodiments to be described in detail below.

Adoption of the concept of a common, agreed capability set definition library permits the remote computer or server application to influence the operation of the client application if not, in some circumstances, entirely control it. The server application is able to do this by, effectively, building a local model of the operation of the client application. Local operation on the model leads to equivalent operations taking effect on the client application.

Capability Set Definition

In order to create a capability set definition library it is first necessary to determine the desired or actual functionality of a device and then define operations which can be invoked on the device or by the device to give effect to that desired functionality. Each operation is, effectively, a "capability" of that device and can form a member of the capability set definition library for that device.

In particular instances the operations will be very basic or primitive capabilities and will be a subset of the total functionality of the device. So, for example, for a display device, one whose primary function is to display, the primitives will be display primitives which, for example, can be defined at the level of a character, the level of a geometrical shape or, alternatively, can be defined mathematically as vectors. These primitives, when invoked and caused to display according to their particular capability will drive the total appearance of the display device.

In alternative embodiments the operations can comprise combinations of primitives which build to form a higher level or more sophisticated or more complex capability.

These operations thus identified. to give effect to the desired functionality are placed into a library so as to form the capability set definition library.

The preferred criteria for selecting members of a capability set definition library are as follows:

1. The innate capability of the client;
2. What the client wants to do or is expected to do with that capability; and
3. The bandwidth of the communications channel and/or other

characteristics of the communications channel such as, for example latency.

Taking into account the criteria the members of a capability set definition library can then be formulated so as to ensure acceptable performance by the client application.

First Preferred Embodiment

With reference to Fig. 1 there is illustrated in block diagram form a distributed system for computer interaction 10 according to a first preferred embodiment of the invention.

In this case the system 10 comprises a device 11 which can communicate via communication channel 12 with a remote computer 13. It is to be understood that both device 11 and remote computer 13 include computing means and memory within them (not shown) which permit the execution of software code. Such code is most usually executed in the environment of an operating system operating on a computing hardware platform. Example current operating systems include Windows 95/98/CE/NT, Unix, Linux. Example current hardware platforms include the Intel series of microprocessors (e.g. Pentium) and the Motorola series of microprocessors. The system of this embodiment is not limited to operation on any particular hardware platform nor under any particular operating system. Indeed it is expected that, most usually, the hardware platform of the remote computer 13 will differ from the hardware platform of the device 11. Similarly it is expected that, mast usually, the operating system of remote computer 13 will differ from the operating system of device 11. All that is required for operation of the system is a compatible communications protocol (an example of which will be given in greater detail below) to permit the transmission of data between remote computer 13 and device 11.

Device 11 is adapted to run or execute via its computing means a client application 14, the client application being able to reference a capability set definition library 15 also loaded or stored on device 11. Whilst device 11 can be almost any kind of device having a computing capability such as, currently but by no means exclusively, a personal computer, a personal digital assistant (PDA), a "palmtop" or hand held (e.g. Windows CE) computer or a controller device operating or assisting the operation of devices such as mobile phones, lifts, industrial controllers (e.g. PLC's) and the like. The device 11 will be exemplified with reference to Fig. 2 as a personal digital assistant (PDA) having a display screen 16 which can communicate with a user via a graphical user interface (GUI)

In the specific example of Figs. 2 and 3 the PDA device 11 is loaded with a GUI client application 14 which is capable of drawing specific objects on screen 16 so as to provide a graphical user interface with a user of PDA device 11. The objects which can form part or all of the graphical user interface are defined in the capability set definition library 15, in this instance being "FLOW PANEL", "TEXT FIELD" and "BUTTON".

The remote computer 13 is adapted to run a control application 17, in this case a "GUI control application", the source code for which is listed in Fig. 3. The source code includes commands which rely on definitions of objects to be found in capability set definition library 18 which, in this example, correspond directly with the object definitions to be found in capability set definition library 15 on PDA device 11. That is, definitions are to be found in the library for "FLOW PANEL", "TEXT FIELD" and "BUTTON" and giving rise to the same graphical constructs as displayed in

Fig. 2 on screen 16.

Lines 16, 17 and 38, 39 define the objects which form the model 20 used by the control application 17.

In this instance, and with reference to earlier discussions in this specification as to the definition of the meaning of "capability set" and "capability set definition library" used in this specification, it is the case in this example that these three library members, FLOW PANEL, TEXT FIELD and BUTTON are each built from a combination of primitives.

These library members are built from selected primitive functions of PDA device 11, specifically

(a) pixel colour control primitive

(b) pointer device tracking primitive

(c) text/character recognition/input primitive.

FLOW PANEL determines screen layout and relies on (a). TEXT FIELD relies on (a) to present an appearance recognisable to a user as a text field. It relies on (b) for selection of text. It relies on (c) for text character input.

BUTTON relies on (a) to present an appearance which the user recognises as a button. It relies on (b) to recognise when the button has been clicked.

With reference to Fig 4 initiation and maintenance of a communication session between device 11 and remote computer 13 over communication channel 12 is illustrated diagrammatically and indicates that a user (not shown) invokes client application 14 which, in turn, creates a remote control protocol (RCP) client 19 which then establishes an RCP connection 20 with an RCP server program 21 running on remote computer 13 which, in turn, communicates with, in this instance, GUI control application 17. It will be observed that a negotiation phase 22 initiates

communication and establish an agreed capability set definition library which both the client application and the control application will reference. Having established this agreement the communication moves to an activity phase 23 whereby predetermined aspects of operation of device 11 are determined by control application 17 with reference to the capability set definition library, which is to say the corresponding capability set definition library 18 for the control application 17 and the capability set definition library 15 for the client application 14.

As will be described in more detail below with reference to Fig. 5 this arrangement permits a user to invoke the FLOW PANEL, TEXT FIELD and BUTTON appearing on screen 16 of the user's PDA device 11 and to have a text message which the user subsequently enters in the TEXT FIELD and a click of the BUTTON to be recognised by control application 17. In practice far more complex interactions will occur.

With reference to Figs. 5A through to Fig. 5K a more detailed description of the interaction between PDA device 11 and remote computer 13 will be given and, in particular, describing in detail the character strings under a particular preferred remote control protocol (RCP) on the communication channel 12:

A PDA device 11, with a set of "simple GUI" capabilities (buttons, text fields, menus) and a simple application called "input" that wants to display a text field and a button to get input from a user (not shown) of the PDA device 11.

Note that this example is slightly contrived for simplicity. In particular, issues about screen layout are avoided here. In this instance, the communication channel 12 comprises a TCP/IP network.

The components of this scenario are outlined in Fig 5A.

Communication over the network follows the following sequential steps:

Figure 5B outlines steps 1-2:

Step 1: The client application 14 running on the PDA 11 establishes a connection to the remote computer 13 running the control application 17 called "input".

Step 2: The server software 24 accepts the incoming connection.

Figure 5C outlines steps 3-8:

Step 3: The server sends "RCP/1.0" to identify that it is using the ROP protocol, version 1.0

Step 4: The client receives the protocol version and. verifies that it is as expected

Step 5: The client sends the application name and capability set version. In this case, that is encoded: "5!input500!" The capability set library, in this instance, comprises three members namely FLOW PANEL, TEXT FIELD, and BUTTON.

Step 6: The server receives the message from the client

Step 7: The server checks the capability set version (500) in this case to make sure the requested application can make use of that capability set. In this case, the check succeeds and the server sends "!" (representing zero) to indicate that the negotiation has succeeded. If initial negotiation indicates a partial overlap in capabilities of the proposed capability sets then a new capability set definition library would be defined comprising either a super set or a sub set of the first proposed libraries so as to ensure that an exact match of capabilities is achieved for both the server and the client.

Step 8: The client receives the confirmation message

from the server

Figure 5D outlines steps 9–10:

Step 9: In this case, there are no application parameters, so the client sends "!" (representing zero) to inform the server of the number of parameters, and the client becomes "active".

Step 10: The server receives the message and also becomes "active".

Figures 5E and 5F outline step 11:

Step 11: The application now takes control of the connection. The first thing this application does is create a model 20 of the user interface which the application is programmed to present on PDA 11 consisting of a text field object and a button object from the simple GUI library of classes. This causes the following two commands to be sent from the server:

"N1!TF!" – creates a text field on the PDA 11 corresponding to the text field object in the model 20 from the capability set with object identity "1".

"N2!BU2!OK" – creates a button on the PDA 11 corresponding to the text field object in the model 20 with identity "2" containing the string "OK"

Figure 5G outlines step 12:

Step 12: The client receives these two commands and creates the text field and button objects on the client as instructed.

Figure 5H outlines steps 13–14:

Step 13: The user can see the objects on the screen and interact with them. When the user enters something into the text field, the client generates the following event:

"E1!2!TC"

Step 14: The server receives this event and the text field object created by the application is notified that its text has changed (i.e. that the client has a new value)

Figure 5I outlines steps 15–16:

Step 15: The user finishes entering the text "hello" into the text field and then clicks the button. This generates the following event sent from the client to the server:

"E2!2!CL" informing the server that the client's button object has been clicked.

Step 16: The server receives this event and the button object created by the application is notified that it was clicked.

Figure 5J outlines steps 17–20:

Step 17: The application responds to this notification by asking the text field what value it holds.

Step 18: The text field knows that the contents held on the server are not up to date because of the earlier "TC" event, (steps 13 and 14) so it requests the text from the client with the following "get text" command:

"M1!2!GT"

Step 19: The client receives this command, and the text field on the client generates the following event:

"E1!9!GT5!hello"

Step 20: The server receives and decodes this event, and the application is informed of the result.

Note that at this point, the user has entered data into a remote application 17 running on remote computer 13. There has been a total of 30 bytes sent from the server remote computer 13 to the client device 11 and 40 bytes sent from

the client device 11 to the server remote computer 13.

Figure 5K outlines step 21:

Step 21: The remote application 17 changes the model 20 by creating a new window object. This change is communicated to the client application by sending the event "N3!DIN5!hello" "M3!3!SVY" with the end result that a new window with the title "hello" is displayed on the GUI interface of the PDA client.

It will be observed that this last step illustrates the active ability of the server application to influence the client application and to do so in a manner which requires only relatively short codes to trigger what can be high level data manipulation or other "high level" activity on or by the client.

In light of the above more detailed account, the example application source code of Fig. 3 will also be described in greater detail corresponding to the usage scenario described above with reference to Figs. 5A–5J.

The source code of GUI control application 17 contains everything that an application programmer would need to know to make use of the system 10. All of the details about the establishment of the communication session and the messages that are passed back and forth between the device and the computer are hidden from applications. With reference to Fig 3:

Starting the application

The application is started by calling the method named "start" (line 22). This method registers the application with the name "input" so that a client can later interact with it. This must be done on the computer before step 1 in the usage

scenario.

Connection establishment

When a client establishes a connection, it is accepted by the server/library code, and after negotiation the application is activated. This results in the server/library code invoking the "newConnection" method (line 35) . This corresponds to step 11 in the usage scenario.

This application creates four objects: a frame, a flow panel, a text field and a button (lines 41—44) The application also registers itself with the button so it can be notified when the button is clicked (line 48) . These are then displayed on the newly established connection (line 51) This results in four "new" commands being sent to the client.

Button click

When the user clicks the button (step 15), the client sends an event to the server, which then passes it to the button. So far, this is invisible to the application. The button then notifies the application (step 16) by calling the "buttonClick" method (line 55). This application then prints out a message on the computer to show that it has been informed that the button was clicked.

Having now described a simplified example of interaction the reader is referred to Annexure A which is a specification for an exemplary remote control protocol (RCP) or Remote Application Protocol and to Annexure B which is a specification for an exemplary capability set suitable for the GUI example previously given. Both of Annexures A and B form part of this specification and are incorporated herein.

Implementation can be in any suitable programming language such as, for example, C++

Second Preferred Embodiment

With reference to Fig. 6 there is shown an implementation of a flexible, distributed system for computer interaction according to a second embodiment of the invention and, in particular, showing more detail of an implementation on a remote computer. In this embodiment like components are numbered as for the first embodiment, except that they are prefixed with the numeral 1. to provide a "100" series of numbers. So, for example, remote computer 13 of the first embodiment becomes remote computer 113 of the second embodiment.

With reference to Fig. 6 there is illustrated in block diagram form a remote computer 113 connected via communications channel 112 to a client device 111.

Within remote computer 113 a capability set definition library 118 is in communication with a plurality of applications including a particular control application 117. The applications run under and with reference to kernel 130 of an operating system.

The relevant parts of a high level architecture of the environment in which applications for system 110 are written are as follows:

- the capability set definition library 118, which is a set of classes that allows control applications 117 to display graphical user interfaces, and includes a server, which sends commands to the client device 111 describing layout of GUI elements in windows and receives and processes events from the client device 111.
- The client application 114, which is a program that runs on device 111 with a display 116 and presents a user

interface. It performs commands as instructed by the server and responds to the user's interactions by sending events to the server. The client application should be

- simple and fast to download and install;
- can be installed as a web browser plug-in for easy access to Kernel services; and
- available across a range of platforms, with implementations for Windows, UNIX MacOS, Palm OS, phones, etc.

Ideally this design has the following characteristics:

- a familiar and simple programming model;
- a simple layout mechanism, easily modeled in a GUI builder application;
- lightweight client-server protocol (usable on 9600 baud serial line);
- applicable to a wide range of devices (display or otherwise);
- integrates handheld devices into business applications;
- has many connections to a server over a modest bandwidth pipe;
- provides a better user experience for Internet services;
- has built in, transparent, standards–compliant security (based on SSL / TLS, X.509 certificates);
- supports coordination of services to provide a distributed business transaction (e.g., Amazon writes code to choose books then links in Fedex for shipping and Visa for payment)
- is an extensible model can take advantage of higher bandwidth if available (plug-ins for streaming audio /

video, canvas support, etc.); and

- has a desktop client that supports web content.

Device Capabilities

The system 110 is designed by concentrating on device capabilities as a central idea. Each device that an application may want to interact with has certain capabilities. It might be able to display a particular set of GUI elements such as buttons and text fields, or it may have a very limited display such as that found on a mobile phone.

To integrate diverse capabilities into a single programming model, the idea is to represent the capabilities of a device in an abstract model on the server. This involves creating a "class" for each of the capabilities a client might have. In the case of a desktop client, the capabilities include:

- the ability to create windows;
- the ability to place GUI elements (buttons, text fields etc.) inside windows; and
- the ability to display web pages.

The preferred capabilities set, comprising a GUI library thus includes classes corresponding to windows, buttons, text fields, and web viewers. Some devices may also have additional capabilities such as handlers for specific types of data, unusual input devices like cameras and microphones, and so on. Each of these can be modeled on the server by a class.

To make use of a device in this model, its primitive operations must be abstracted into a capability set. The same device may be abstracted in different ways. For example, a device with a bitmapped display can be abstracted either as a

primitive such as a simple frame buffer that allows applications to set the color of each pixel, or it can be abstracted in terms of higher level GUI constructs composed of multiple primitives such as buttons, text fields and tree controls. The level of abstraction will determine the bandwidth requirements and the application interface

A capability set is implemented by a set of classes (called the capability set model), and a set of client classes (called the client capability set) A device may implement more than one capability set, and an application may be written to make use of more than one capability set. To establish an active connection, the client and server must agree on which capability set to use for the remainder of the connection. Reaching this agreement is the purpose of negotiation.

The Protocol

A typical session is started as follows:

- a client connects to the server and requests a particular application
- the client and server negotiate about the capabilities to be used for this session
- the client and server agree on a versioned set of classes, including a basic capability set (e.g. Desktop PC, Palm device, robot)

Once client and server agree on the classes representing the basic capability set, the operations are simple:

- construct an object of a particular class
- send a message to an object
- forget about an object (delete)

- quit (close this connection)
- "browse" to another application and/or another server
- check whether a particular extension supported (e.g., can the client handle a particular kind of data, or is a particular input device available, etc.)

The "objects" constructed on the client are very different from the objects being manipulated by server applications. On the client side the objects have direct control of the corresponding capability of the underlying client device. On the server side, they are only lightweight models.

All messages are class-specific (i.e., not specified by the basic protocol), so the implementation of each server class and the client class must match for the system to work. This can be achieved for example by requiring that all interfaces including extended capability classes are versioned, so that the set of classes that the client and server agree on match.

Collecting multiple applications into a single user interface

A device 111 with the client application 114 has an extra capability, the ability to make further connections to other servers. That is, a recursive capability which can be supported in exactly the same way as "ordinary" capabilities described thus far.

This particular capability is interesting, however, because it provides further new ways to use the client program:

- one application can cause the client to make a sub-connection to another application

- e.g. of use: bookshop implements book choosing, then delegates to Fedex for shipping and Visa for payment

With reference to Fig. 7 possibilities resulting from invoking of multiple connections, for example over the internet, are illustrated diagrammatically including the recursive scenario referred to above. The end result is a powerful system which, when tuned appropriately, can provide rapid response times as perceived by a user operating the devices, 11, 111, 212. notwithstanding that a significant level of overhead is initiated in the process on remote computers 13, 113, 213.

Third Preferred Embodiment

With reference to Fig. 8 an example is given where the client device is a mobile telephone 21 and the capabilities (or individual functions) which work together to define the total functionality of the mobile phone can include:

1. Receive call
2. Initiate call
3. Display character
4. Voice recognise word

These defined capabilities can be thought of as primitives, each of which defines only a small portion of the total functionality of the mobile phone device but which, when collected together, provide, in totality, useful capabilities grouped into the device commonly known as a mobile telephone.

In this instance it is desirable to select the capabilities with a view to minimising the amount of update information that will need to be passed between the client

device and remote computer having the control application on it.

Typically the aim will be not to define the capabilities at a very high level which would require the passing of many parameters between the client and the remote computer in order to update the status of that capability or to implement that capability.

Equally it will typically not be desirable to define the capabilities at too low a level where the number of capabilities needed to be invoked and/or kept track of in order to provide meaningful functionality on the client device will be too large which, in turn, will also contribute to an unnecessarily large amount of traffic passing between the client device and the remote computer.

Fourth Preferred Embodiment

With reference to Fig. 9 an example is illustrated wherein the client device is a lift controller 22 arid its capabilities are broken down into the following members:

1. Go to floor number
2. Open door
3. Close door

Fifth Preferred Embodiment

With reference to Fig. 10 there is illustrated a particular implementation of the arrangement of Fig. 6 and wherein the client is a laptop computer 23 which is loaded with the capability set definitions by means of a CD-ROM 24 having thereon the necessary definitions.

In an alternative form the definitions can be downloaded via the internet from a host site.

THE CAPABILITY SET DEFINITION LIBRARY EXPRESSED AS A MODEL

As has been discussed earlier in the specification with particular reference to the definition of "capability set" the effect of the creation of a capability set and a capability set definition library relevant to any given application can also be thought of as the creation of a "model" of the application or at least some aspects of it for the particular purpose, inter alia, of co-ordinating operation of the model on a client device with the operation of a corresponding model on the server device so as to minimize the amount of data which needs to pass between the client and the server for operation of the application on the client.

Descriptions of a further three preferred embodiments now follow wherein the term "model" is used rather than "capability set" or "capability set definition library".

Sixth Preferred Embodiment

A method and system for facilitating a computer controlling an interaction with a device using a model of the device is disclosed in Fig. 11. The system and method involve a device asking for a connection to a computer. Upon connecting the device transmits identifying information and instructions to the computer. The computer then creates a complete model of the device based on this information, the model essentially being an abstraction of the device in the memory of the computer. The computer then determines the behaviour of the model based on its identifying information and instructions and on the applications and information that the computer contains. After determining the behaviour the computer changes the model to reflect this and stores the

appropriate information to be transmitted to the device in the model. The computer then transmits this information to the device. The device presents the information, interacting of or with it and generating a response. This response is then recorded on the device and transmitted back to the computer. The computer in turn changes its model to reflect the response and again determines the behaviour of the device, changes the model again to reflect the newly determined behaviour and transmits its response back to the device so that the device can again interact with it.

Fig. 11 shows one example of a method and system for facilitating a computer controlling an interaction with a device using a model according to this 6th embodiment. Referring to Fig. 11 it can be seen that the device 204 transmits a response reaction 210 to the computer 201. The computer 201 determines the behaviour 207 of the Device by creating a Device Model 202 and determining what applications and information 203 can be used by the device model 202 leading to the generation of a format 208. The format 208 is then incorporated into the device model 202 and transmitted as a question action 211 to the device 203 for presentation and interaction 205 with the format 209. The device 204 records the presentation and interaction 205 and its response and then transmits the response reaction 210 back to the computer 201 that again determines the behaviour 207 of the device 204.

Seventh Preferred Embodiment

In this seventh preferred embodiment there is described with reference to Fig. 12 a method of implementing a client user interface for an input/output device which is adapted to be interconnected with a software application, the method

comprising the steps of: defining a first series of classes that simulate a model of the functional capabilities of the input/output device; defining a second series of classes corresponding to the programs running on the device; wherein to perform an action, the software application interacts with the first series of classes and upon an interaction, the first series of classes sends output command events corresponding to the action to the second series of classes running on the device; and the second series of classes sends input command events to the first series of classes which translates the input command events into corresponding input actions and notifies the software application of the input action.

The input/output device can comprise for example one of a palm computer, a mobile phone, a handheld computer or a desktop computer.

In this seventh preferred embodiment with reference to Fig. 12, a model is introduced for a software application interacting with user interface type devices. The system proceeds by utilizing a software representation of a model of the device with the application program interacting with the model. The arrangement of this preferred embodiment is illustrated in Fig. 12 wherein the application software program 301 interacts with a software model 302 of the interaction device with the interaction with the software model 302 being subseqeuntly automatically translated into operations on a corresponding interaction device 303. The construction of the arrangement of Fig. 12 can proceed by utilization of a series of library classes that allow the application 301 to display, for example, graphical user interfaces etc. The classes themselves send commands to a client program running on the interaction device 303. The

commands describe the layout of the graphical user interface elements and also receive and process events such as keyboard events from the interaction device 303. The client program 303A is a program that runs on the device and displays and presents a user interface to its user. The client program 303Amerely needs to be able to interact with the model of the interactive device and perform commands as instructed by the model and responds to the users interactions by sending events to the server. In this manner, when it is desired to utilize a different device, all that really needs to be written is the form of interaction with a corresponding model of the new device. This design allows for the following improved characteristics:

- a familiar and simple programming model;
- a simple layout mechanism, easily modelled in a GUI builder application;
- lightweight client-server protocol (usable on 9600 baud serial lines);
- application to a wide range of devices (display or otherwise);
- integrates handheld devices into business applications;
- many connections to a server are possible on a modest bandwidth interconnect;
- provides a better user experience for Internet services;
- can have built in, transparent, standard-compliant security (based on SSL/TLS, X.509 certificates);
- supports coordination or services to provide a distributed business transactions;
- is an extensible model that can take advantage of higher bandwidth if available (plug-ins for streaming

audio/video, canvas support, etc.); and

- has a desktop client that supports web content.

The mechanism of construction concentrates on device capabilities as a central idea. Each device that an application may want to interact with will have certain capabilities. It might be able to display a particular set of GUI elements such as buttons and text fields, or it may have a very limited display such as that found on a mobile phone.

To integrate diverse capabilities into a single programming model in a simple manner, the capabilities of the device are presented abstract model on the server. This involves creating a software "class" of reach of the capabilities a client might have. For example, in the case of a desktop client, the capabilities include:

- the ability to create windows;
- the ability to place GUI elements (buttons, text fields, etc.) inside windows; and
- the ability to display web pages.

The GUI library thus includes classes corresponding to windows, buttons, text fields, and web displayers. Some devices may also have additional capabilities such as handlers for specific types of data, unusual input devices like cameras and microphones, and so on. Each of these can be modelled on the server by a class. A typical session can then be based around a protocol and started as follows:

- a client program 3 connects to the server and request a particular application;
- the client and server negotiate about the capabilities

to be used for a current session

- the client and server agree on a versioned set of classes, including a basic capability set (eg. Desktop PC, Palm device, robot)

Once the client and server agree on the classes representing the basic capability set, the operations on behalf of the server software application are simple:

- construct an object of each required particular class
- send appropriate messages to the object
- delete an object
- quit (close the current connection)
- "browse" to another application andlor another server
- check whether a particular extension supported (eg. Can the client handle a particular kind of data, or is this a particular input device available, etc.)

The "objects" constructed on the client can be very different from the object being manipulated by the application. On the client side the objects have direct control of the corresponding physical capability of the underlying client device. On the server side, they are only lightweight models.

All messages are class-specific in that they are not specified by the basic protocol, so the implementation of each server class and the client class must match for the system to work.. This can be achieved by requiring that all interfaces including extended capability classes are versioned, so that the set of classes that the client and server agree on match.

A device 303 with the client can have an extra

capability, under this model. This is the ability to make connection to different servers. This suggests new ways to use the client program:

- one application can cause the client to make a sub-connection to another application as required.

By utilising this alternative model a simplified form of constructing user interfaces is presented where a client builds its own slave object model for interaction with by software applications wishing to utilize the model. In this way a universal client can be provided which is able to be utilised in many different contexts.

It would be appreciated by a person skilled in the art that numerous variations and/or modifications may be made to the present invention as shown in the specific embodiments without departing from the spirit or scope of the invention as broadly described. The present embodiments are, therefore, to be considered in all respects to be illustrative and not restrictive.

INDUSTRIAL APPLICATIONS

The system according to various embodiments of the invention can be applied on almost any device which has a computing capability and the ability to communicate with other computing devices, whether on a one to one basis or as part of a much wider network of computing devices. The system can be utilised to enhance the capabilities of personal computing devices such as PDA's, PC's hand held computers and the like. It can also be used to enhance the capability of computer enabled industrial devices including controllers of various kinds such as lift controllers, programmable logic controllers and the like.

ANNEXURE A

"BULLANT" KERNEL REMOTE APPLICATION PROTOCOL SPECIFICATION PAPER VERSION 1.2

Contents

1. DOCUMENT PURPOSE 3

2. BACKGROUND 3

3. ARCHITECTURE 3

4. THE PROTOCOL 3

4.1 OVERVIEW 3
4.2 SECURITY 4
4.3 ENCODING AND DECODING VALUES 4
4.4 CONNECTION ESTABLISHMENT 5
4.5 NEGOTIATION 5
4.6 STATE TRANSITIONS 6
4.7 COMMANDS 7
4.8 EVENTS 8

5. RAP URL DEFINITION 9

6. DOCUMENT CONTROL 9

6.1 MANIFEST 9
6.2 HISTORY 10

1. Document Purpose

This document describes the Remote Application Protocol (RAP) (previously known as the Remote Control Protocol (RCP)) in sufficient detail to make it possible to implement clients and servers that communicate using the protocol. This document is intended for developers and assumes the background of the Remote (previously known as the Antenna) White Paper.

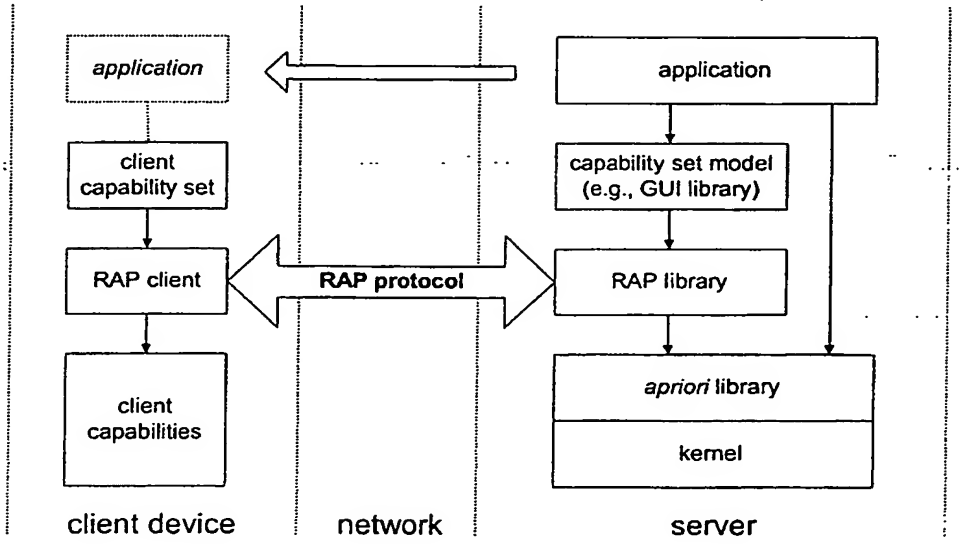
2. Background

The Remote Application Protocol forms the basis of the Remote architecture. It allows application programs to control external devices by constructing a logical model of the device and manipulating the model. The protocol is responsible for keeping the application's model and the device's state synchronized, so that changes to one are reflected in the other.

This document assumes that RAP is implemented over some communication network that provides reliable communication, such as TCP/IP. Thus no error correction is built into this protocol. No other assumptions are made about the underlying network communication except when the client is instructed to make a new connection to a server, in which case the new server's address must be encoded in the protocol. It is assumed that the address can be encoded as a string of text.

3. Architecture

The basic architecture described in the Remote White Paper is decomposed further here to make the role of the Remote Application Protocol clearer.



4. The Protocol

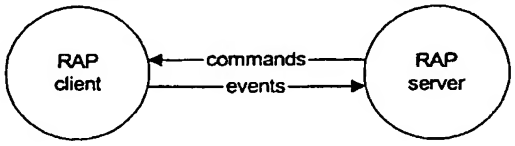
4.1 Overview

A successful Remote Application Protocol connection goes through the following phases:

1. connection establishment
2. negotiation
3. activity
4. closure

Connections are established by the client, so the programming model is that applications register themselves with the "RAP server" task, which is part of the RAP library. The RAP server then waits for incoming connections from clients. When a client connects, the RAP server performs the negotiation, and if successful passes the connection on to the application requested by the client.

Once the negotiation is completed and the application takes over the connection, communication is simple:



4.2 Security

The server can be configured to accept connections on more than one "port", and can be set up to expect encrypted connections on some ports and unencrypted (plain text) connections on other ports (as with web servers). This allows privacy to be guaranteed using a lower level protocol such as SSL or TLS.

An encrypted connection may provide extra capabilities to the server including user authentication using digital certificates. That can be implemented above this level of the protocol by treating authentication similarly to the other capabilities of the device.

4.3 Encoding and Decoding Values

4.3.1 Boolean Values

Boolean values are either true or false. They are transmitted as either the character "Y" for true or "N" for false. Such values are referred to as `bool` in the remainder of this document.

4.3.2 Integer Values

Integers are transmitted in decimal form followed by an exclamation mark. Numbers thus match the regular expression:

```
("-")?<digit>*"!"
```

(this means: an optional minus sign followed by zero or more digits followed by an exclamation mark). For example, to send the integer 314, following would be sent:

```
314!
```

Such values are referred to as `int` for the remainder of this document. If no digits are sent (i.e., just an exclamation mark), then the decoded value is zero. Thus zero is encoded by a single character ("!").

4.3.3 Fixed length strings

Fixed length strings are used where the number of characters in the string is known in advance, such as literal strings that are part of the protocol. Such strings are encoded as follows (for a string of length "n"):

```
<character>[n]
```

(this means exactly "n" characters). The fixed length string "RAP/1.1" would be transmitted as:

```
RAP/1.1
```

Such a value is referred to as `string[n]` for the remainder of this document, where "n" is the (fixed) length.

4.3.4 Variable length strings

If the length of the string is not known in advance, the value is transmitted as follows (again for a string of length "n"):

```
<n : int><character>[n]
```

(this means: the length of the string encoded as an integer followed by the "n" characters). For example, the string "hello" would be transmitted:

```
5!hello
```

Such values are referred to just as `string` for the remainder of this document. Note that the empty `string` is encoded as a single character ("!").

4.3.5 Raw bytes

An sequence of raw bytes (such as the bytes encoding an image) as transmitted similarly to variable length strings:

```
<n : int><byte>[n]
```

(this means: the number of bytes encoded as an integer followed by the bytes). Such values are referred to as `raw` for the remainder of this document. Note that the empty `raw` is encoded as a single character ("!").

4.4 Connection Establishment

The client is responsible for establishing new connections. To create a RAP connection, the client must know the following parameters:

1. the address of the server
2. the port that the RAP server is listening on for incoming connections
3. the name of the application being requested
4. any parameters to be passed to the application

These parameters are encoded as a list of `<name, value>` pairs, and called the *connection properties*. The pre-defined names are:

1. "host"
2. "port"
3. "application"

Any other names are assumed to be application parameters. These pairs are typically stored in an ASCII text file, and a MIME type is associated with the file. This allows the client to work as a web browser plug-in. These parameters can also be sent over an established connection by the server to instruct the client to open a new connection using the "open" command described below.

4.5 Negotiation

Negotiation provides the means by which many different kinds of devices can be incorporated into applications using RAP. The model is that each device can provide one or more *capability sets*. A capability set is embodied in a set of SoftBlocks classes and a corresponding set of *slave* classes on the client. The SoftBlocks classes are used to construct a model of the device in a SoftBlocks application, and the slave classes run on the client and make the client device's capabilities available via the protocol

Each capability set to be supported is identified by a unique integer. For example, the initial implementation of the Remote that makes desktop GUI capabilities available to SoftBlocks applications uses the identifier 1000.

During negotiation, the client indicates to the server the capability sets it supports and the server indicates which capability sets the requested application is prepared to use. Negotiation *succeeds* if the client and server can agree to use a particular capability set. The result of a successful negotiation is that the application knows which set of SoftBlocks classes can be used to control the device, and the device knows what commands to expect from the server.

To ensure that negotiation terminates, the protocol specifies that the client and server must suggest candidate capability sets in *decreasing* order. Thus, a client should first suggest its highest numbered capability set and if that is not acceptable, the server should suggest the next highest numbered set, and so on.

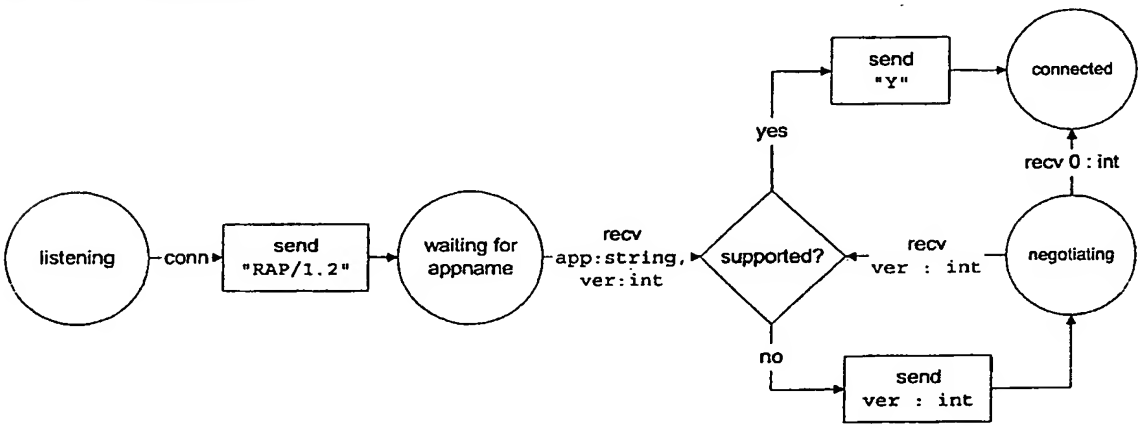
This implies that identifiers should be assigned to capability sets on the basis of how rich they are: the richer the set of capabilities, the higher its identifier. Different versions of the one capability set should also be numbered in order of versions (assuming that it is more desirable to use later version). Of course, there is no linear scale onto which all devices can be placed, so (for example) robots can not be compared with Palm devices. The capability set identifiers needs to be carefully assigned to avoid confusion.

4.6 State transitions

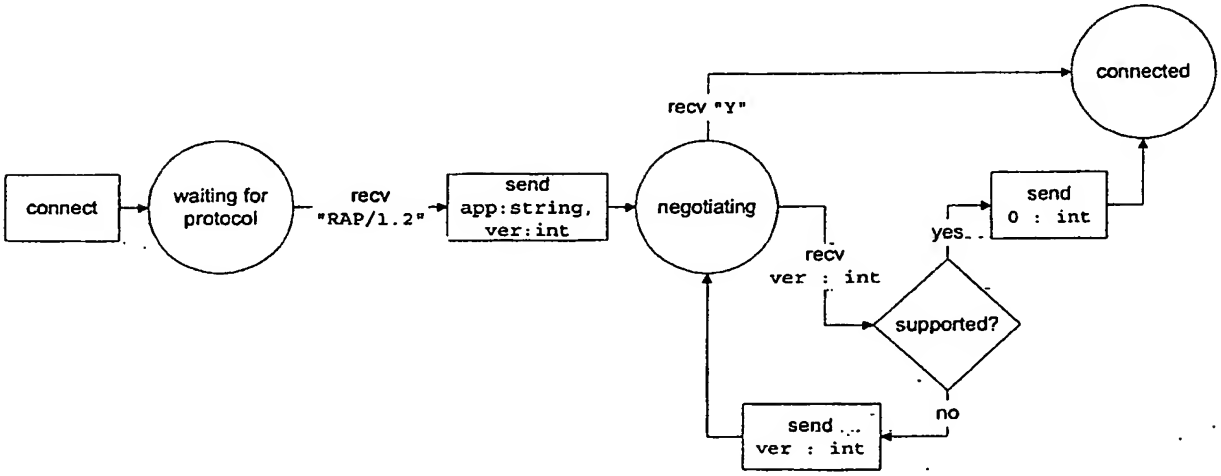
The following two state transition diagrams summarize and clarify the way the protocol works. The first diagram shows the state transitions that the server undergoes, and the second shows the client states. The notation consists of:

- circles represents states;
- boxes represent actions;
- arrows represents changes in states;
- labels on arrows give conditions under which that transition can occur; and
- diamonds represent decisions.

4.6.1 Server negotiation

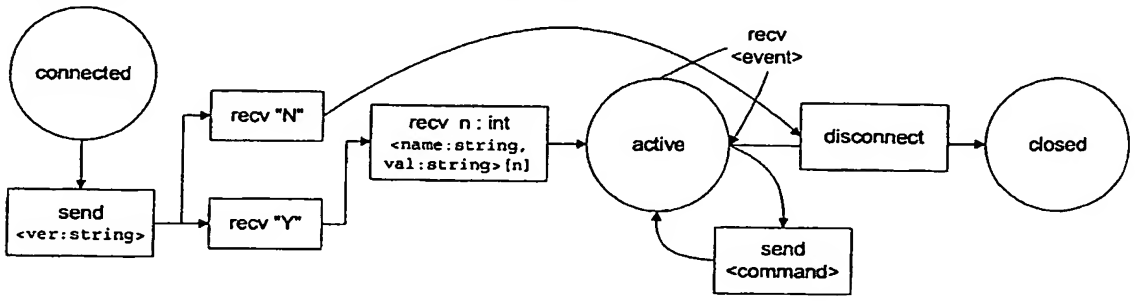


4.6.2 Client negotiation



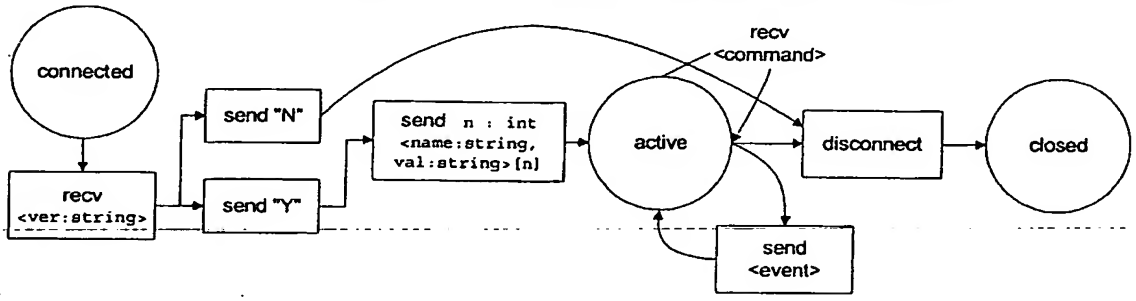
4.6.3 Server activity

Once negotiation is complete, the client receives from the server the library version string. The client is able to compare this with a built in value and decide whether to proceed or disconnect. On deciding to proceed, the client sends any application parameters, and the connection becomes *active*. At this point, the server sends commands and receives events until the client disconnects.



4.6.4 Client activity

The client's state transitions after negotiation have completed are a mirror image of the server's.



4.7 Commands

The application interacts with the device by creating objects from a library that models the capability set agreed on by the client and server during negotiation. The application can then call methods on those objects to create a model of how it wants the device to operate. RAP causes the device to act on behalf of the application in accordance with the model the application builds. The protocol uses *commands* to cause the device to change state based on changes to the application's model.

4.7.1 New Object

The server can cause the client to construct a new object based on a class in the capability set. The server assigns a unique integer to identify each new object. All classes in a capability set are identified by a string of two characters (e.g., "BU" for button).

The format of a "new" command is:

```
"N"<id : int, classid : string[2], constructor : raw>
```

This instructs the client to construct a new object with identifier `id` from the class identified by `classid`, and to use the class-specific data in the `constructor` to initialize the new object. The contents and format of the data in the `constructor` is determined by the class and is outside the scope of this document.

The server refers to the new object by its identifier in all future commands. These identifiers are simple integers, and no guarantee is made by the protocol about the order or range in which the server assigns identifiers except that they must be positive (non-zero) integers.

4.7.2 Send Message

Once the client has constructed an object in response to a "new" command, the server can interact with that object by sending it a message. This is done with the "send message" command:

```
"M"<id : int, message : raw>
```

Upon receiving this message, the client finds the object identified by `id` and passes it the `message`. As with the constructor of a new command, the contents and format of the data in the `message` is determined by the class of the object and is outside the scope of this document

4.7.3 Forget Object

The server can instruct the client to "forget" about an element using a "forget" command. Its format is simply:

```
"F"<id : int>
```

After this command is sent, the server will not send `id` in any future commands until the identifier is recycled by another "new" command.

4.7.4 Quit

Once an application no longer needs a connection (e.g., because it has completed) it uses the "quit" command to inform the client that the session has ended.

```
"Q"<message : string>
```

If the application completes "normally", the `message` should be empty, and the client need not display anything. If the message is not empty, however, the client can assume that something went wrong, and the message should be processed somehow by the client (e.g., by displaying a dialog in the case of a GUI client).

4.7.5 Open a connection

An application can also instruct the client to open another connection to some other application. The new connection can either replace the existing one or operating simultaneously with the existing connection. The new connection is defined by its parameters.

```
"O"<newconn : bool><url : string>
```

The client responds by either opening a new connection or re-using the existing connection as indicated by the `newconn` parameter. The `<url>` specifies the new connection. It is specified in section 5 below.

4.7.6 Query whether an extension is supported

In addition to the classes that make up a capability set, a client may support *extensions*. These are extra classes that are added to a capability set at runtime, analogously to web browser "plugins". Each extension is assigned a unique identifying string. To use such an extension, the application must be compiled with the corresponding *extension model* class. The server then sends this command to determine whether the extension is supported.

```
"S"<classid : string[2]><extensionid : string>
```

The client should respond with an "extension supported" event. The server will use `classid` to refer to this class in future "new" commands.

4.8 Events

4.8.1 Normal Events

The client informs the server about changes to its state using *events*. A typical example might be a user interface device that generates events based on user actions. RAP can be used to send such events from the client objects to the corresponding objects on the server. The RAP layer on the server translates these into method calls on the objects in the application's model, and the application can then respond to these events in whatever way the programmer wants.

Note that there is no enforced correspondence between commands and events. Some commands from the server may result in events from the client, and vice versa, but many may not. Either commands or events may be sent asynchronously. This must be taken into account when designing capability sets.

The format of an event is:

```
"E"<id : int><body : raw>
```

As with messages sent by the application, the contents and format of the event body are defined by the class of the object sending or receiving the event.

4.8.2 Exceptions

Unusual events that do not occur in normal device operation can be indicated by an "exception" event. The format is almost identical to normal events:

```
"X"<id : int><body : raw>
```

Again the meaning depends on the class involved. There is a special case for exceptions, however, which is that the client application itself (rather than a specific object) can raise an exception by using an `id` of zero.

4.8.3 Extension supported response

Upon receipt of an "extension supported" command, the client sends this response. It indicates to the server whether the requested extension is supported by this client.

```
"R"<classid : string[2]><supported : bool>
```

If `supported` is true, the server can construct new objects using `classid` in future "new" commands.

5. RAP URL Definition

The RAP URL is based upon RFC 1738 although RAP is not currently a registered scheme. The format is:

```
[rap://]hostname[:port][/appname][?name[=val][&name[=val]] ... ]

note hostname is the default
  myhost          == rap://myhost/default
  myhost/test     == rap://myhost/test
and no host defaults to localhost e.g.
  /test           == rap://localhost/test
  /               == rap://localhost/default
```

Refer to RFC 1738 for details, but note that the following characters are reserved:

```
";" | "/" | "?" | ":" | "@" | "&" | "="
```

and to include any of the above it is necessary to escape them using the %<hex><hex> notation.

6. Document Control

6.1 Manifest

Name	Remote Application Protocol
File	Client Products/RAP-specification.doc
Kind	Specification
Status	Draft
Classification	Company Use Only
Distribution	Restricted
Responsibility	mjc

ANNEXURE B

"BULLANT" KERNEL DESK TOP GUI

CAPABILITY SET VERSION 1.14

Contents

1. DOCUMENT PURPOSE 4

2. BACKGROUND 4

3. EVOLUTION 4

4. CONSTANTS 4

4.1 ALIGNMENT 4

5. WINDOWS 5

5.1 WINDOW 5
5.2 FRAME 5
5.3 DIALOG 6

6. ELEMENTS 6

6.1 ELEMENT 6
6.2 BUTTON 7
6.3 CANVAS 8
6.4 CHECKBOX 10
6.5 COMBOBOX 10
6.6 IMAGEBUTTON 11
6.7 LISTBOX 12
6.8 PROGRESSBAR 13
6.9 RADIO BUTTON 13
6.10 RADIO GROUP 13
6.11 SLIDER 14
6.12 STATICIMAGE 14
6.13 STATICTEXT 15
6.14 TEXTAREA 15
6.15 TEXTFIELD 16
6.16 TREECONTROL 17
6.17 WEBVIEWER 18

7. LAYOUT 18

7.1 PANEL 18
7.2 BORDERPANEL 19
7.3 COLUMNPANEL 20
7.4 FLOWPANEL 20
7.5 GRIDPANEL 21
7.6 IMAGEPANEL 21
7.7 SCROLLPANEL 22
7.8 SPLITTERPANEL 22
7.9 TABPANEL 23

8. MENUS 23

8.1 MENUELEMENT 23
8.2 MENUITEM 23
8.3 MENU 24
8.4 MENUBAR 24
8.5 POPUPMENU 25

9. MISCELLANEOUS CLASSES 25
9.1 IMAGE 25
9.2 GIF IMAGE 26
9.3 JPEG IMAGE 26
9.4 PNG IMAGE 26
10. SECURITY CLASSES 26
10.1 PASSWORDAUTHENTICATOR 26
10.2 PASSWORDCHOOSER 27
10.3 CERTIFICATEAUTHENTICATOR 28
10.4 CERTIFICATECREATOR 28
10.5 STATEMENTSIGNER 29
11. FILE OPERATIONS 29
11.1 FILEDOWNLOADER 29
11.2 FILEUPLOADER 30
12. MESSAGE CODES APPENDIX 31
12.1 INTRODUCTION 31
12.2 MESSAGE CODES 31
12.3 REBUILDING THE TABLE 34
13. DOCUMENT CONTROL 35
13.1 MANIFEST 35
13.2 HISTORY 35

1. Document Purpose

This document describes the Desktop GUI library capability set communication in sufficient detail for an engineer to implement the Desktop GUI library model in Bullant and a GUI library client on a specific device. It does not attempt to explain the semantics of all the events and messages – that can be found in the "GUI Programmer's Guide".

2. Background

The Desktop GUI library is a *capability set* as defined by the Remote White Paper. To define a capability set, it is sufficient to list the classes, and for each class to describe:

- its class identifier;
- the class(es) it inherits from;
- the constructor;
- the messages that the server sends to the client; and
- the events that the client sends to the server.

This Desktop GUI library uses the same encoding of values that is described in the Remote Control Protocol specification, and this document uses the same notation for representing parameters. One additional convention is used: if a class name appears (the convention of using capital letters to start class names is adopted here) as the type of a value, then the integer identifier of an object of that class is the value that is encoded.

3. Evolution

As the DesktopGUI has evolved, the issue of gracefully evolving additions the capability set needed to be addressed. The choice of defining a new capability set is always an option, but this can be disruptive and not necessarily a suitable response. A compromise position has been adopted which allows additions to be made the capability set that will allow an old version of the DesktopGUI library to communicate with a newer Remote (the version checking feature of RAP will warn the user when the DesktopGUI library is newer than the Remote). The rules are thus:

- new messages may be added
- extra arguments may be added to messages, but the type or ordering of arguments may not change
- defaults are defined for the cases when arguments are missing

4. Constants

Some messages involve choices from a range of values. In cases where these are used in several messages, the meaning of the values is defined here.

4.1 Alignment

Alignment of elements within panels or text within columns is one of:

0 = top or left

1 = centered

2 = bottom or right

3 = stretched

The "stretched" value indicates that an element will be stretched to the width or height of the container. It has no meaning for text in columns. Values of this type are referred to as `Alignment` below.

5. Windows

5.1 Window

CLASS IDENTIFIER

none (abstract class)

INHERITS

none

MESSAGES

Get size – request to get the size of a window

```
"GS"
```

Set content – sets the element that is inside a window

```
"SC"<content : Element>
```

Set size – sets the size of the window (in pixels or proportion of screen?)

```
"SS"<width : int, height : int>
```

Set title – sets the title string in the title bar of the window

```
"ST"<title : string>
```

Set visible – causes the window to become visible / invisible

```
"SV"<visible : bool>
```

EVENTS

Window closed – sent to indicate that the window has been closed, either by the user or programmatically

```
"WC"
```

Window size – reply to a get size message. Contains the width and height of the window

```
"WS"<width : int, height : int>
```

Window resized – sent to indicate that the window has been resized

```
"WR"<width : int, height : int>
```

5.2 Frame

CLASS IDENTIFIER

```
"FR"
```

INHERITS

Window

CONSTRUCTOR

```
<title : string><initState : int><trayMin : bool><<icon :
Image><tooltip : string>>[?icon != NULL]
```

where `initState` has the following values:

- 0 – restored
- 1 – minimised

- 2 – maximised

and `trayMin` represents whether or not to minimise the frame to the system tray, `icon` is an image to display in the caption and system tray, and `tooltip` is text to display when the mouse is over the system tray.

MESSAGES

Set menu bar – sets the menu bar displayed in the top of the frame.

```
"SM"<menubar : MenuBar>
```

Set frame state – sets the frame to one of the states as described above for `initState`.

```
"FS"<state : int>
```

Set icon – sets the icon to display in the frame's caption, and in the system tray if the frame is being minimised to the system tray. Also sets the tooltip to associate with the image in the system tray.

```
"SI"<icon : Image><tooltip : string>
```

Minimise to tray – sets whether or not the frame should be minimised to the system tray.

```
"MT"<toTray : bool>
```

EVENTS

Frame state changed – the frame state has changed to one of the states as described above for `initState`.

```
"FS"<state : int><width : int><height : int>
```

5.3 Dialog

CLASS IDENTIFIER

```
"DI"
```

INHERITS

Window

CONSTRUCTOR

```
<parent : Window, modal : bool, title : string>
```

MESSAGES

Set modal – causes the window to become modal or modeless

```
"SM"<modal : bool>
```

EVENTS

none

6. Elements

6.1 Element

CLASS IDENTIFIER

none (abstract class)

INHERITS

none

MESSAGES

Set enabled – causes the element to become enabled / disabled

```
"EN"<enabled : bool>
```

Request focus – causes the window to become focused (i.e. receive keyboard input)

```
"RF"
```

Set font

```
"SF"<face  : string, type : int, style : int, size : int>
```

where:

- `face` specifies the font names to try in a comma separated list. It may be null (zero length).
- `type` is one of: 0 = normal (user selected), 1 = serif, 2 = sans serif, 3 = fixed width
- `style` is one of: 0 = regular, 1 = bold, 2 = italic, 3 = bold italic, 4 = underline, 5 = bold underline, 6 = italic underline, 7 = bold italic underline
- `size` is from 1-100, where 10 is the user selected default. i.e. *size is always relative to the default*

Set foreground colour and set background colour

```
"FC"<colourSpec : string>

"BC"<colourSpec : string>
```

where the colour specification is a hexadecimal string: rrggbb[aa], with colour components from 0 to 255 for red, green, blue and alpha (opacity). For the alpha component, 0 indicates completely transparent, and FF (255) indicates completely opaque. If the alpha component is omitted, FF is assumed. Note that clients may not support the full range of colours, and may not support alpha transparency at all.

Set popup menu

```
"SP"<menu : PopupMenu>
```

EVENTS

Lost focus – the focus has changed to another element

```
"LF"
```

Gained focus – the focus has changed to this element

```
"GF"
```

6.2 Button

CLASS IDENTIFIER

```
"BU"
```

INHERITS

Element

CONSTRUCTOR

```
<text : string>
```

MESSAGES

Set content – sets the text or image contained in the button

```
"SC"<text : string>
```

EVENTS

Button clicked – sent when the user clicks the button

```
"CL"
```

6.3 Canvas

The client should maintain an off-screen buffer where the drawing commands are performed, and just copy the pixels between that buffer and the on-screen element. This keeps network bandwidth to a minimum.

CLASS IDENTIFIER

```
"CV"
```

INHERITS

Element

CONSTRUCTOR

```
<width : int, height : int>
```

where the sizes are in pixels.

MESSAGES

Clear a rectangular area

```
"CL"<x : int, y : int, width : int, height : int>
```

Copies (bitblts) a rectangular area

```
"CO"<fromX : int, fromY : int, toX : int, toY : int, width : int,
height : int>
```

Clear pointer regions – canvas will no longer send "RE" or "RL" events

```
"CR"
```

Draw an arc

```
"DA"<x : int, y : int, width : int, height : int, startAngle : int,
sweepAngle : int>
```

where the angles are in tenths of a degree. A start angle of zero means the middle of the right-hand side of the bounding rectangle.

Draw an image

```
"DI"<image : Image, imageX : int, imageY : int, x, y, width : int,
height : int>
```

Draw a line

```
"DL"<x1 : int, y1 : int, x2 : int, y2 : int>
```

Draw a polygon

```
"DP"<numPoints : int><x : int, y : int>[numPoints]
```

Draw a rectangle

```
"DR"<x : int, y : int, width : int, height : int>
```

Draw an interpolated spline between the specified points

```
"DS"<numPoints : int><x : int, y : int>[numPoints]
```

Draw text – text is clipped and aligned to the bounding box.

```
"DT"<text : string, x : int, y : int, width : int, height : int,
hAlign : Alignment, vAlign : Alignment>
```

Fill an arc

```
"FA"<x : int, y : int, width : int, height : int, startAngle : int,
sweepAngle : int>
```

where the angles are in tenths of a degree.

Fill a polygon

```
"FP"<numPoints : int><x : int, y : int>[numPoints]
```

Fill a rectangle

```
"FR"<x : int, y : int, width : int, height : int>
```

Create new pointer region – client should now send "RE" and "RL" events for this region. The new region is appended onto the end of the region list, and it's index in the list is used to identify it.

```
"NR"<x : int, y : int, width : int, height : int>
```

Get text size using the currently selected font – client should respond with a "TS" event

```
"TS"<queryID : int, text : string>
```

Set line width in pixels (initially 1)

```
"LW"<pixelWidth : int>
```

Remove region – delete a previously created region. Higher region indices are decremented.

```
"RR"<regionIndex : int>
```

Set size – set new size in pixels. Old contents will be clipped to new size.

```
"SS"<width : int, height : int>
```

Update the canvas – ensure that any pending drawing commands are displayed on the screen

```
"UP"
```

Update a rectangle – ensure that any pending drawing commands in a rectangle are displayed on the screen

```
"UR"<x : int, y : int, width : int, height : int>
```

Set a clipping region – restrict the drawing area to this rectangle. This area is by default the size of the canvas.

```
"SR"<x : int, y : int, width : int, height : int>
```

Reset the clipping region back to default (the size of the canvas)

```
"RC"
```

Set XOR drawing mode on or off (initially off)

```
"XO"<useXOR : bool>
```

EVENTS

Pointer down

```
"DO"<x : int, y : int, clicks : int>
```

Pointer drag

```
"DR"<x : int, y : int>
```

Pointer region entered – sent when the mouse is moved into a pointer region

```
"RE"<regionIndex : int>
```

Pointer region left – sent when the mouse is moved out of a pointer region

```
"RL"
```

Resized – called when the canvas is resized on the client

```
"RS"<width : int, height : int>
```

Pointer up

```
"UP"<x : int, y : int>
```

Text size response to a "TS" query

```
"TS"<queryID : int, width : int, height : int>
```

6.4 Checkbox

CLASS IDENTIFIER

```
"CH"
```

INHERITS

Element

CONSTRUCTOR

```
<state : bool, text : string, image : Image>
```

where image may be null.

MESSAGES

Set content – sets the text or image contained in the button

```
"SC"<text : string, image : Image>
```

Set state

```
"SS"<state : bool>
```

EVENTS

State changed

```
"SC"<newState : bool>
```

6.5 ComboBox

CLASS IDENTIFIER

```
"CO"
```

INHERITS

Element

CONSTRUCTOR

```
<editable : bool, text : string>
```

MESSAGES

Get text – request for the client to send a "GT" event containing its text

```
"GT"
```

Insert item – inserts the item at the specified index

```
"IN"<index : int, itemText : string>
```

Remove item

```
"RE"<index : int>
```

Set contents – sets all items at once

```
"SC"<numItems : int><itemText : string>[numItems]
```

Select item – cause the numbered item to be selected and its text displayed in the combo box

```
"SI"<index : int>
```

Set text – set the string in the text field (note: this is ignored by non-editable combo boxes)

```
"ST"<text : string>
```

Update item – resets the text of the item at the specified index

```
"UP"<index : int><itemText : string>
```

EVENTS

Item selected – sent when the user selects an item

```
"SE"<index : int>
```

Get text event – sent in reply to a "GT" command. Contains the new text in the combo box.

```
"GT"<text : string>
```

Return pressed – sent when the user presses return in the combo box. May also send new value of the text stored in the combo box.

```
"RE"<hastext : bool><text : string>[hastext]
```

Text changed event – sent when the user changes the text from the last value known to the server

```
"TC"
```

6.6 ImageButton

CLASS IDENTIFIER

```
"BI"
```

INHERITS

Button

CONSTRUCTOR

```
<normal : Image, rollover : Image, pressed : Image, focused : Image,
drawBorder : bool><disabled : Image>?
```

The disabled image is optional and defaults to null, in which case, the disabled state is indicated by a greyed out version of the normal image.

MESSAGES

Set content – sets the images contained in the button

```
"SC"<normal : Image, rollover : Image, pressed : Image, focused :
Image><disabled : Image>?
```

The disabled image is optional and defaults to null, in which case, the disabled state is indicated by a greyed out version of the normal image.

EVENTS

none apart from the inherited Button event(s)

6.7 ListBox

CLASS IDENTIFIER

```
"LB"
```

INHERITS

Element

CONSTRUCTOR

```
<multiSelection : bool, multiColumn : bool><<numColumns :
int><colTitle : string, align : Alignment, isImage :
bool>[numColumns]>[?multiColumn]
```

If `multicolumn` is false, then `numColumns` is set to 1, and `isImage` is set to false.

MESSAGES

Insert item – inserts a new item at the specified index

```
"IN"<index : int><<text : string>[?not col.isImage]<image :
Image>[?col.isImage]>[numColumns]
```

Remove item

```
"RE"<index : int>
```

Set contents – sets all items at once

```
"SC"<numItems : int><<text : string>[?not col.isImage]<image :
Image>[?col.isImage]>[numColumns * numItems]
```

Select/Unselect item(s)

```
"SE"<index : int, selected : bool>
```

Set selection – set which items are selected

```
"SS"<numItems : int><index : int>[numItems]
```

Update item – resets the value of an item

```
"UP"<index : int><<text : string>[?not col.isImage]<image :
Image>[?col.isImage]>[numColumns]
```

EVENTS

Item executed – called when an item is "executed" (i.e. double-clicked)

```
"EX"<index : int>
```

Item selected – called when an item is "selected" (i.e. single-clicked)

```
"SE"<index : int>
```

Item unselected – called when an item is "unselected" in a multiple selection list box

```
"UN"<index : int>
```

6.8 ProgressBar

CLASS IDENTIFIER

```
"PR"
```

INHERITS

Element

CONSTRUCTOR

```
<minvalue : int, maxvalue : int, value : int>
```

MESSAGES

Set range

```
"SR"<minValue : int, maxValue : int>
```

Set Value

```
"SV"<value : int>
```

EVENTS

none

6.9 Radio Button

CLASS IDENTIFIER

```
"RB"
```

INHERITS

Element

CONSTRUCTOR

```
<text : string, image : Image>
```

where image may be null

MESSAGES

Set content – sets the text or image contained in the button

```
"SC"<text : string, image : Image>
```

EVENTS

none

6.10 Radio Group

Note that a radio group is not an element in itself. The buttons that make up a group can be laid out in any way, but events are funneled through the radio group object.

CLASS IDENTIFIER

```
"RG"
```

INHERITS

none

CONSTRUCTOR

```
<numButtons : int><button : RadioButton>[numButtons]
```

MESSAGES

Add a button to the group

```
"AB"<button : RadioButton>
```

Remove a button from the group

```
"RE"<button : RadioButton>
```

Select a button from the group (the first button is selected by default when the group is first created)

```
"SE"<button : RadioButton>
```

EVENTS

Button selected

```
"SE"<button : RadioButton>
```

6.11 Slider

CLASS IDENTIFIER

```
"SL"
```

INHERITS

Element

CONSTRUCTOR

```
<horizontal : bool, minvalue : int, maxvalue : int, value : int>
```

MESSAGES

Set range

```
"SR"<minValue : int, maxValue : int>
```

Set value

```
"SV"<value : int>
```

EVENTS

Value changed – sent when the user changes the value (i.e., drags then releases)

```
"VC"<value : int>
```

6.12 StaticImage

CLASS IDENTIFIER

```
"SI"
```

INHERITS

Element

CONSTRUCTOR

```
<image : Image>
```

MESSAGES

Set content – sets the image contained in this static image element

```
"SC"<image : Image>
```

EVENTS

none

6.13 StaticText

CLASS IDENTIFIER

```
"ST"
```

INHERITS

Element

CONSTRUCTOR

```
<text : string>
```

MESSAGES

Set content – sets the text contained in this static text element

```
"SC"<text : string>
```

EVENTS

none

6.14 TextArea

CLASS IDENTIFIER

```
"TE"
```

INHERITS

Element

CONSTRUCTOR

```
<text : string, rows : int, columns : int, wordwrap : bool>
```

MESSAGES

Append text – adds text to the end of the current contents of the text field

```
"AT"<text : string>
```

Get text – request an update from the client. The client should reply with a "GT" event

```
"GT"
```

Set editable – cause the text area to be editable or not

```
"SE"<editable : bool>
```

Set text – change the text store in the text area to a new value

```
"ST"<text : string>
```

Set size – set the size of the text area in rows and columns

```
"SS"<rows : int, columns : int>
```

Word wrap – turn word wrap on or off

```
"WW"<wordwrap : bool>
```

Text select – Select text in the range [selectFrom, selectTo) and move the cursor to the end of the selection

```
"TS"<selectFrom : int, selectTo : int>
```

EVENTS

Get text event – sent in reply to a "GT" command. Contains the new text in the text area.

```
"GT"<text : string>
```

Text changed event – sent when the user changes the text from the last value known to the server

```
"TC"
```

6.15 TextField

CLASS IDENTIFIER

```
"TF"
```

INHERITS

Element

CONSTRUCTOR

```
<text : string, width : int, masked : bool>
```

MESSAGES

Append text – adds text to the end of the current contents of the text field

```
"AT"<text : string>
```

Get text – request an update from the client. The client should reply with a "GT" event

```
"GT"
```

Set editable – cause the text area to be editable or not

```
"SE"<editable : bool>
```

Set masked – cause the text field to be masked or not. Note that this command also empties the text field

```
"SM"<masked : bool>
```

Set text – change the text store in the text field to the new value

```
"ST"<text : string>
```

Set width – set the size of the text field columns

```
"SW"<width : int>
```

Text select – Select text in the range [selectFrom, selectTo) and move the cursor to the end of the selection

```
"TS"<selectFrom : int, selectTo : int>
```

EVENTS

Get text event – sent in reply to a "GT" command. Contains the new text in the text field.

```
"GT"<text : string>
```

In case the textfield is set to "hashed" or new text is MD5-hashed before it is send, the length of the original text is also send:

```
"GT"<text : string><hashedTextLength : int>
```

Return pressed – sent when the user presses return in the text field. May also send new value of the text stored in the text field.

```
"RE"<hastext : bool><text : string>[hastext]
```

Text changed event – sent when the user changes the text from the last value known to the server

```
"TC"
```

6.16 TreeControl

Note that tree controls are special in the sense that the items that they display are also like objects from the point of view of the connection. Each tree control must maintain its own mapping of item identifiers to tree control item objects.

CLASS IDENTIFIER

```
"TC"
```

INHERITS

Element

CONSTRUCTOR

```
""
```

MESSAGES

Forget (delete) an item – it's item identifier can then be reused

```
"FO"<item : TreeControlItem>
```

Insert item as a child at an index within an existing item

```
"IN"<parent : TreeControlItem, index : item : TreeControlItem>
```

Create a new container item (as the last child in its parent, if non-null)

```
"NC"<parent : TreeControlItem, id : int, text : string, closedicon :
Image, openicon : Image>
```

Create a new leaf item (as the last child in its parent, if non-null)

```
"NL"<parent : TreeControlItem, id : int, text : string, icon : Image>
```

Open or collapse an item

```
"OP"<item : TreeControlItem, opened : bool>
```

Remove an item

```
"RE"<parent : TreeControlItem, index : int>
```

Set contents – set the root of the tree of items

```
"SC"<root : TreeControlItem>
```

Select item

```
"SE"<item : TreeControlItem>
```

Update a container item

```
"UC"<id : TreeControlItem, text : string, closedicon : Image,
openicon : Image>
```

Update a leaf item

```
"UL"<id : TreeControlItem, text : string, icon : Image>
```

EVENTS

Item executed – called when an item is "executed" (i.e. double-clicked)

```
"EX"<item : TreeControlItem>
```

Item opened – called when an item is expanded or collapsed

```
"OP"<item : TreeControlItem, open : bool>
```

Item selected – called when an item is "selected" (i.e. single-clicked)

```
"SE"<item : TreeControlItem>
```

6.17 WebViewer

CLASS IDENTIFIER

```
"WV"
```

INHERITS

Element

CONSTRUCTOR

```
<isURL : bool, contents : string>
```

MESSAGES

Show a HTML string – interpret the content string as HTML and display it.

```
"SH"<content : string>
```

Show HTML from a URL

```
"SU"<url : string>
```

Print the contents of a WebViewer

```
"PR"
```

EVENTS

Link selected – sent when the user selects a URL in the displayed HTML.

```
"SE"<url : string>
```

7. Layout

7.1 Panel

CLASS IDENTIFIER

none (abstract class)

INHERITS

Element

MESSAGES

Set border type from the following:

- None 0
- Line 1
- Raised 2
- Lowered 3
- Etched 4

```
"BT"<borderType : int>
```

Layout the panel – ensure that any changes to the child list is reflected in the display

```
"LA"
```

Set gap between elements in pixels – ***not necessarily meaningful for all panel types.***

```
"SG"<hGap : int, vGap : int>
```

EVENTS

none

7.2 BorderPanel

CLASS IDENTIFIER

```
"BP"
```

INHERITS

Panel

CONSTRUCTOR

```
""
```

MESSAGES

Set bottom element

```
"SB"<bottom : Element>
```

Set center element

```
"SC"<center : Element>
```

Set left element

```
"SL"<left : Element>
```

Set right element

```
"SR"<right : Element>
```

Set top element

```
"ST"<top : Element>
```

EVENTS

none

7.3 ColumnPanel

CLASS IDENTIFIER

```
"CP"
```

INHERITS

Panel

CONSTRUCTOR

```
<rowAlign : Alignment, numCols : int><align : Alignment, stretch :
int>[numCols]
```

MESSAGES

Set the row alignment

```
"AR"<rowAlign : Alignment>
```

Inserts a column at the given index into the list of columns

```
"IC"<index : int, align : Alignment, stretch : int>
```

Inserts a child at the given index into the list of children

```
"IN"<index : int, child : Element>
```

Removes the column at the given index from the list of columns

```
"RC"<index : int>
```

Removes the child from the list of children

```
"RE"<child : Element>
```

Set all children at once

```
"SC"<numChildren : int><child : Element>[numChildren]
```

EVENTS

none

7.4 FlowPanel

CLASS IDENTIFIER

```
"FP"
```

INHERITS

Panel

CONSTRUCTOR

```
<horizontal : bool, vAlign : Alignment, hAlign : Alignment>
```

MESSAGES

Set the horizontal alignment

```
"AH"<hAlign : Alignment>
```

Set the vertical alignment

```
"AV"<vAlign : Alignment>
```

Inserts a child at the given index into the list of children

```
"IN"<index : int, child : Element>
```

Removes the child from the list of children

```
"RE"<child : Element>
```

Set all children at once

```
"SC"<numChildren : int><child : Element>[numChildren]
```

Set the layout direction

```
"SD"<horizontal : bool>
```

EVENTS

none

7.5 GridPanel

CLASS IDENTIFIER]

```
"GP"
```

INHERITS

Panel

CONSTRUCTOR

```
<rows : int, columns : int>
```

MESSAGES

Inserts a child at the given index into the list of children

```
"IN"<index : int, child : Element>
```

Removes the child at the given index from the list of children

```
"RE"<index : int >
```

Set all children at once

```
"SC"<numChildren : int><child : Element>[numChildren]
```

EVENTS

none

7.6 ImagePanel

CLASS IDENTIFIER

```
"PI"
```

INHERITS

Panel

CONSTRUCTOR

```
<child : Element, image : Image>
```

MESSAGES

Set the child in the image panel

```
"SC"<child : Element>
```

Set the background image to be tiled behind the child

```
"SI"<image : Image>
```

EVENTS

none

7.7 ScrollPanel

CLASS IDENTIFIER

```
"SP"
```

INHERITS

Panel

CONSTRUCTOR

```
<child : Element>
```

MESSAGES

Sets the child in the scroll panel

```
"SC"<child : Element>
```

EVENTS

none

7.8 SplitterPanel

CLASS IDENTIFIER

```
"pp"
```

INHERITS

Panel

CONSTRUCTOR

```
<horizontal : bool, child1 : Element, child2 : Element, prop1 : int,
prop2 : int>
```

where `prop1` and `prop2` are the proportions in which the panel will initially be divided.

MESSAGES

Set first child

```
"S1"<child1 : Element>
```

Set second child

```
"S2"<child2 : Element>
```

Set direction

```
"SD"<horizontal : boolean>
```

Set the proportions of the division

```
"PN"<prop1 : int, prop2 : int>
```

EVENTS

Panel resized – sent when the splitter is moved

```
"SM"<prop1 : int, prop2 : int>
```

7.9 TabPanel

CLASS IDENTIFIER

```
"TP"
```

INHERITS

Panel

CONSTRUCTOR

```
""
```

MESSAGES

Inserts a child into the tab panel

```
"IN"<index : int, child : Element, name : string, image : Image>
```

Removes a child from the tab panel

```
"RE"<child : Element>
```

Set contents – sets all children at once

```
"SC"<numChildren : int><child : Element, name : string, image :
Image>[numChildren]
```

Select page – raises the specified tab page

```
"SE"<index : int>
```

EVENTS

Tab selected – sent when the user raises one of the tabs

```
"SE"<index : int>
```

8. Menus

8.1 MenuElement

Note: this class is an abstract class representing any element that can be added to a Menu, including MenuItems, Menus and menu separators (represented by a null MenuElement).

CLASS IDENTIFIER

none (abstract class)

INHERITS

none

8.2 MenuItem

Note that a separator is indicated by a null menu item (i.e. the zero handle)

CLASS IDENTIFIER

```
"MI"
```

INHERITS

MenuElement

CONSTRUCTOR

```
<text : string><enable : bool>?
```

Enable is optional and defaults to true.

MESSAGES

Set text

```
"ST"<text : string>
```

Enable item

```
"EI"<enable : bool>
```

EVENTS

Item selected – sent when the user selects this item. If the menu item belongs to a PopupMenu, then owner is the RAP Id of the GUIElement for which the popup was invoked. If the menu belongs to a MenuBar, then owner has a value of 0.

```
"SE"<owner : GUIElement>
```

8.3 Menu

CLASS IDENTIFIER

```
"ME"
```

INHERITS

MenuElement

CONSTRUCTOR

```
<name : string, numItems : int><item : MenuElement>[numItems]
```

MESSAGES

Insert item : item can be either an ordinary menu item, a separator (represented by null) or another menu (to create cascading menus).

```
"IN"<index : int, item : MenuElement>
```

Remove item

```
"RE"<index : int>
```

Set name – sets the name displayed at the top of the menu

```
"SN"<name : string>
```

EVENTS

none

8.4 MenuBar

CLASS IDENTIFIER

```
"MB"
```

INHERITS

none

CONSTRUCTOR

```
<numMenus : int><menu : Menu>[numMenus]
```

MESSAGES

Insert a menu at the specified index

```
"IN"<index : int, menu : Menu>
```

Remove a menu

```
"RE"<index : int>
```

Set help menu – note that this menu should not be added with "IN". Use a null menu to remove the help menu.

```
"SH"<menu : Menu>
```

EVENTS

none

8.5 PopupMenu

CLASS IDENTIFIER

```
"PM"
```

INHERITS

none

CONSTRUCTOR

```
<numItems : int><item : MenuElement>[numItems]
```

MESSAGES

Insert item : item can be either an ordinary menu item, a separator (represented by null), or a menu (to create cascading menus)

```
"IN"<index : int, item : MenuElement>
```

Remove item

```
"RE"<index : int>
```

EVENTS

none

9. Miscellaneous classes

9.1 Image

CLASS IDENTIFIER

none (abstract class)

INHERITS

none

CONSTRUCTOR

```
<id : string><size : int><modTime : string>
```

where:

- `id` is some symbolic name for the image including an MD5 hash
- `size` is the size of the image in bytes
- `modTime` is the modification date and time of the image file

Note: if the client has an image cached with the same `id`, `size` and `modTime` from the same server, it can send the "IR" event immediately.

MESSAGES

Data – add data to end of image data

```
"DA"<data : raw>
```

EVENTS

Image ready – sent when the client has enough information to display the part or all of the image

```
"IR"<width : int><height : int>
```

Need data – sent once by the client after the constructor is parsed if the client doesn't have the image

```
"ND"
```

9.2 GIF Image

Note that none of the image formats have any extra methods or special constructors. The subclasses each handle an image data format.

CLASS IDENTIFIER

```
"IG
```

INHERITS

Image

9.3 JPEG Image

CLASS IDENTIFIER

```
"IJ
```

INHERITS

Image

9.4 PNG Image

CLASS IDENTIFIER

```
"IP"
```

INHERITS

Image

10. Security Classes

10.1 PasswordAuthenticator

CLASS IDENTIFIER

```
"PA"
```

INHERITS

none

CONSTRUCTOR

```
<parent : Window>
```

Note: may be null

MESSAGES

Show – make the window visible, send an event when done

```
"SH"
```

EVENTS

Cancelled – window closed without password being entered

```
"CA"
```

Password entered

```
"PE"<username : string, passwordHash : string>
```

where passwordHash is the MD5 hash of the password

10.2 PasswordChooser

CLASS IDENTIFIER

```
"PC"
```

INHERITS

none

CONSTRUCTOR

```
<parent : Window>
```

Note: may be null

MESSAGES

Show – make the window visible, send an event when done

```
"SH"
```

Set user name – if set, the user name is displayed but is not editable

```
"SU"<username : string>
```

EVENTS

Cancelled – window closed without password being entered

```
"CA"
```

Password entered

```
"PE"<name : string, passwordHash : string>
```

where passwordHash is the MD5 hash of the (confirmed) password

10.3 CertificateAuthenticator

This window allows the user to choose a certificate and if the user can decrypt the associated private key, renegotiates the current secure connection using the certificate.

CLASS IDENTIFIER

"CA"

INHERITS

none

CONSTRUCTOR

<parent : Window>

Note: may be null

MESSAGES

Show – make the window visible, send an event when done

"SH"

EVENTS

Cancelled – window closed without password being entered

"CA"

Connection renegotiated – certificate details can be recovered from there

"CR"

10.4 CertificateCreator

This window allows the user to enter details and create a certificate request (including private key). The private key is then encrypted using a password selected by the user. The certificate request is sent in an event, and the server can then respond with an issued certificate.

CLASS IDENTIFIER

"CC"

INHERITS

none

CONSTRUCTOR

<parent : Window>

Note: may be null

MESSAGES

Issue certificate

"IC"<certificate : raw>

where certificate is a PEM-formatted X.509 certificate.

Show – make the window visible, send an event when done

"SH"

EVENTS

Cancelled – window closed without password being entered

`"CA"`

Certificate request

`"CR"<request : raw>`

where the request is a PEM-formatted X.509 certificate request.

10.5 StatementSigner

This window displays a statement (piece of HTML text) to the user and prompts the user to sign the statement. It can only be used on Certified connections (i.e., ones for which the user has been authenticated using a certificate). If the user opts to sign the statement, the signature is sent to the server in an event.

CLASS IDENTIFIER

`"SS"`

INHERITS

none

CONSTRUCTOR

`<parent : Window, title : string, statement : string>`

Note: `parent` may be null, `statement` is HTML text.

MESSAGES

Show – make the window visible, send an event when done

`"SH"`

EVENTS

Cancelled – window closed without statement being signed

`"CA"`

Statement signed

`"SS"<signature : raw>`

where `signature` is the result of signing the statement with the user's private key.

11. File operations

11.1 FileDownloader

This window allows the user to receive data and save it to a file on their local machine

CLASS IDENTIFIER

`"FD"`

INHERITS

none

CONSTRUCTOR

`<parent : Window, title : string, filename : string, size : int>`

Note: `parent` may not be null, and `filename` is a simple filename without a path, which may be null to force the user to choose a name.

MESSAGES

Show – make the window visible, send an event when done

```
"SH"
```

File data –a chunk of file data

```
"FD"<data : raw, more : bool>
```

where `more` is true for all but the last chunk of the file.

EVENTS

Cancelled – window closed without choosing a filename

```
"CA"
```

File chosen – data to follow

```
"FC"
```

File download finished - unsuccessfully

```
"FE"
```

File download finished - successfully

```
"FS"
```

11.2 FileUploader

This window allows the user to choose a file on their local machine and sends the file data to the server.

CLASS IDENTIFIER

```
"FU"
```

INHERITS

none

CONSTRUCTOR

```
<parent : Window, title : string, extension : string, chunksize :
int>
```

Note: `parent` *may not be null, and* `extension` *is a simple filename extension without wildcards (e.g., "doc" for Word files), which may be null to indicate that any file type is acceptable. The last parameter indicates the size of chunks to use when uploading the file. It may be zero to indicate that the entire file should be sent in a single event.*

MESSAGES

Show – make the window visible, send an event when done.

```
"SH"
```

EVENTS

Cancelled – window closed without a filename being chosen

```
"CA"
```

File chosen – data to follow

```
"FC"<filename : string, size : int>
```

where `filename` is the base name of the file (without directories), and `size` is the file size in bytes.

File data - a chunk of file data

```
"FD"<data : raw, more : bool>
```

where `more` is true for all but the last chunk of the file.

CLAIMS

1. A distributed system for computer interaction; said system including
a device having computing means therein and
a remote computer located or locatable remotely from said device, said device arranged to execute a client application which makes reference to a client capability set definition library located on said device; said remote computer arranged to execute a control application which makes reference to a corresponding capability set definition library located on said remote computer.

2. The system of Claim 1 wherein said system operates sequentially in the following steps:
(a) said client application on said device initiates communication with said control application on said remote computer for the purpose of establishing a communications session between said device and said remote computer;
(b) said client application and said control application agree on mutual use of said capability set definition library;
(c) said control application and said client application maintaining communication during said communication session whereby predetermined aspects of operation of said device are determined by said control application with reference to said capability set definition library.

3. A distributed client/server system for at least a first device and
a first remote computer
said first device arranged to execute a client

application which makes reference to an at least first client capability set driver;

said at least first remote computer arranged to execute a control application which makes reference to an at least first capability set definition library;

said system operating sequentially in the following steps:

(a) said client application on said device initiates communication with said control application on said remote computer for the purpose of establishing a communications session;

(b) said client application and said control application agree on a capability set definition library and a corresponding client capability set driver for use during said communication session;

(c) said control application and said client application maintaining communication during said communication session whereby predetermined aspects of operation of said device are determined by said control application with reference to said selected capability set definition library and said corresponding selected client capability set driver

4. The system of Claim 2 wherein said capability set definition library contains a set of definitions of predetermined, selected capabilities of said device.

5. The system of Claim 4 wherein said predetermined capabilities are primitive functions of the device.

6. The system of Claim 4 wherein said capabilities are selected in order to minimise traffic during said communications session.

7. The system of claim 2 wherein said communications session includes the communicating of update information

pertinent to the status of individual ones of said capabilities invoked with the aid of said capability set definition library.

8. The system of claim 7 wherein said communications session occurs over a network link.

9. A distributed system for computer interaction; said system including

a device having computing means therein and

a remote computer located or locatable remotely from said device, communication means whereby bi—directional communication is established between said device and said remote computer;

said device arranged to execute a client application; said remote computer arranged to execute a control application.

10. The system of Claim 9 wherein said device makes reference to a client capability set definition library located on said device.

11. The system of Claim 10 wherein said control application makes reference to said capability set definition library.

12. The system of Claim 10 wherein said capability set definition library resides on both said device and said remote computer.

13. The system of Claim 9 wherein said system operates sequentially in the following steps:

(a) said client application on said device initiates communication with said control application on said remote computer for the purpose of establishing a communications session between said device and said remote computer;

(b) said client application and said control

application agree on mutual use of said capability set definition library;

(c) said control application and said client application maintaining communication during said communication session whereby predetermined aspects of operation of said device are determined by said control application with reference to said capability set definition library.

14. The system of Claim 13 wherein said capability set definition library contains a set of definitions of predetermined selected capabilities of said device.

15. The system of Claim 14 wherein said predetermined capabilities are primitive functions of the device.

16. The system of Claim 14 wherein said capabilities are selected in order to minimise traffic during said communication session.

17. The system of claim 14 wherein said communication session includes the communicating of update information pertinent to the status of individual ones of the capabilities invoked with the aid of said capability set definition library.

18. In a distributed system for computer interaction; said system including

a device having computing means therein and

a remote computer located or locatable remotely from said device, said device arranged to execute a client application; said remote computer arranged to execute a control application; a method of operating said system according to the following steps:

(a) making available a capability set definition library on both said device and said remote computer;

(b) said client application on said device initiates communication with said control application on said remote computer for the purpose of establishing a communications session between said device and said remote computer;

(c) said client application and said control application agree on mutual use of said capability set definition library;

(d) said control application and said client application maintaining communication during said communication session whereby predetermined aspects of operation of said, device are determined by said control application with reference to said capability set definition library.

19. The system of Claim 9 wherein said remote computer builds a model of the state of said client application so as to interact with said device by interacting with said model.

20. The system of Claim 19 wherein the said model is continuously updated so as to reflect current status of said client application on said device

21. The system of Claim 20 wherein said device is arranged to execute said client application with reference to a client capability set definition library located on said device and said remote computer is arranged to execute said control application by making reference to a corresponding capability set definition library located on said remote computer.

22. The system of Claim 21 wherein said capability set definition library comprises a plurality of capability definitions, each of said definitions determined with reference to the desired functionality of said device.

23. The system of Claim 22 wherein each of said capability definitions comprises a primitive function of said device.

24. The system of Claim 22 wherein each of said capability definitions comprises a combination of primitive functions of said device.

25. A distributed client/server system for at least a first device and

a first remote computer;

communication means whereby bi—directional communication is established between said device and said remote computer;

said first device arranged to execute a client application which makes reference to an at least first client capability set driver;

said at least first remote computer arranged to execute a control application which makes reference to an at least first capability set definition library; said system operating sequentially in the following steps:

(a) said client application on said device initiates communication with said control application on said remote computer for the purpose of establishing a communications session;

(b) said client application and said control application agree on a capability set definition library and a corresponding client capability set driver for use during said communication session;

(c) said control application and said client application maintaining communication during said communication session whereby predetermined aspects of operation of said device are determined by said control application with reference to said selected

capability set definition library and said corresponding selected client capability set driver;

said system arranged to communicate codes over said communication means which make reference to capability definitions in said capability set definition library thereby to control the volume of traffic on said communication means.

26. A distributed system for computer interaction; said system including

a device having computing means therein and

a remote computer located or locatable remotely from said device, said device arranged to execute a client application which makes reference to a client capability set definition library or model located on said device; said remote computer arranged to execute a control application which makes reference to a corresponding capability set definition library or corresponding model located on said remote computer.

27. The system of Claim 26 wherein said system operates sequentially in the following steps:

(a) said client application on said device initiates communication with said control application on said remote computer for the purpose of establishing a communications session between said device and said remote computer;

(b) said client application and said control application agree on mutual use of said capability set definition library or model;

(c) said control application and said client application maintaining communication during said communication session whereby predetermined aspects

of operation of said device are determined by said control application with reference to said capability set definition library or model.

28. A distributed client/server system for at least a first device and

a first remote computer

said first device arranged to execute a client application which makes reference to an at least first client capability set driver or first model;

said at least first remote computer arranged to execute a control application which makes reference to an at least first capability set definition library or corresponding first model;

said system operating sequentially in the following steps:

(a) said client application on said device initiates communication with said control application on said remote computer for the purpose of establishing a communications session;

(b) said client application and said control application agree on a capability set definition library or model and a corresponding client capability set driver or corresponding model for use during said communication session;

(c) said control application and said client application maintaining communication during said communication session whereby predetermined aspects of operation of said device are determined by said control application with reference to said selected capability set definition library or model and said corresponding selected client capability set driver or corresponding model.

29. The system of Claim 27 wherein said capability set definition library or model contains a set of definitions of predetermined, selected capabilities of said device.

30. The system of Claim 29 wherein said predetermined capabilities are primitive functions of the device.

31. The system of Claim 29 wherein said capabilities are selected in order to minimise traffic during said communications session.

32. The system of claim 27 wherein said communications session includes the communicating of update information pertinent to the status of individual ones of said capabilities invoked with the aid of said capability set definition library or model.

33. The system of claim 32 wherein said communications session occurs over a network link.

34. A distributed system for computer interaction; said system including

a device having computing means therein and

a remote computer located or locatable remotely from said device, communication means whereby bi–directional communication is established between said device and said remote computer;

said device arranged to execute a client application;

said remote computer arranged to execute a control application.

35. The system of Claim 34 wherein said device makes reference to a client capability set definition library or model located on said device.

36. The system of Claim 350 wherein said control application makes reference to said capability set definition library or model.

37. The system of Claim 35 wherein said capability set definition library or model resides on both said device and said remote computer.

38. The system of Claim 34 wherein said system operates sequentially in the following steps:

(a) said client application on said device initiates communication with said control application on said remote computer for the purpose of establishing a communications session between said device and said remote computer;

(b) said client application and said control application agree on mutual use of said capability set definition library or model;

(c) said control application and said client application maintaining communication during said communication session whereby predetermined aspects of operation of said device are determined by said control application with reference to said capability set definition library or model.

39. The system of Claim 35 or 36 wherein said capability set definition library or model contains a set of definitions of predetermined selected capabilities of said device.

40. The system of Claim 39 wherein said predetermined capabilities are primitive functions of the device.

41. The system of Claim 39 wherein said capabilities are selected in order to minimise traffic during said communication session.

42. The system of claim 39 wherein said communication session includes the communicating of update information pertinent to the status of individual ones of the capabilities invoked with the aid of said capability set

definition library.

43. In a distributed system for computer interaction; said system including

a device having computing means therein and

a remote computer located or locatable remotely from said device, said device arranged to execute a client application; said remote computer arranged to execute a control application; a method of operating said system according to the following steps:

(a) making available a capability set definition library or model on both said device and said remote computer;

(b) said client application on said device initiates communication with said control application on said remote computer for the purpose of establishing a communications session between said device and said remote computer;

(c) said client application and said control application agree on mutual use of said capability set definition library or model;

(d) said control application and said client application maintaining communication during said communication session whereby predetermined aspects of operation of said, device are determined by said control application with reference to said capability set definition library or model.

44. The system of Claim 34 wherein said remote computer builds a model of the state of said client application so as to interact with said device by interacting with said model.

45. The system of Claim 44 wherein the said model is continuously updated so as to reflect current status of

said client application on said device

46. The system of Claim 45 wherein said device is arranged to execute said client application with reference to a client capability set definition library located on said device and said remote computer is arranged to execute said control application by making reference to a corresponding capability set definition library located on said remote computer.

47. The system of Claim 46 wherein said capability set definition library comprises a plurality of capability definitions, each of said definitions determined with reference to the desired functionality of said device.

48. The system of Claim 47 wherein each of said capability definitions comprises a primitive function of said device.

49. The system of Claim 47 wherein each of said capability definitions comprises a combination of primitive functions of said device.

50. A distributed client/server system for at least a first device and

a first remote computer;

communication means whereby bi-directional communication is established between said device and said remote computer;

said first device arranged to execute a client application which makes reference to an at least first client capability set driver or model;

said at least first remote computer arranged to execute a control application which makes reference to an at least first capability set definition library or corresponding model;

said system operating sequentially in the following

steps:

(a) said client application on said device initiates communication with said control application on said remote computer for the purpose of establishing a communications session;

(b) said client application and said control application agree on a capability set definition library or model and a corresponding client capability set driver or corresponding model for use during said communication session;

(c) said control application and said client application maintaining communication during said communication session whereby predetermined aspects of operation of said device are determined by said control application with reference to said selected capability set definition library or model and said corresponding selected client capability set driver or corresponding model;

said system arranged to communicate codes over said communication means which make reference to capability definitions in said capability set definition library or model thereby to control the volume of traffic on said communication means.

51. A method and system for facilitating a computer controlling an interaction with a device using a model of the device:

- The said device asking for a connection to the said computer
- The said computer creating a complete model being an abstraction of the said device on the said computer

- The said computer determining and imposing the appropriate behaviour information of the said device model abstraction on the said computer
- The said computer changing the said device model abstraction on the said computer to reflect the said imposed appropriate behaviour information of the said device model abstraction on the said computer
- The said computer transmitting the said imposed appropriate behaviour information to the said device
- The said device presenting the said imposed appropriate behaviour information
- The said device transmitting appropriate response behaviour information to the said computer
- The said computer receiving the said appropriate response behaviour information from the said device and transmitting determined and imposed appropriate behaviour information as a response.

52. The method of claim 51 wherein said step of the said device asking for a connection to the said computer includes the step of the said device determining appropriate instructions about what it is, its current state and what it can do.

53. The method of claim 52 wherein said step of the said device determining appropriate instructions about what it is, its current state and what it can do includes the step of the said device determining what applications or information contained on the said computer the said device requires.

54. The method of claim 53 wherein said step of the said

device determining what applications or information contained on the said computer the said device requires includes the step the said device turning its appropriate instructions into appropriate information that can be conveyed to the said computer via a connection medium.

55. The method of claim 54 wherein said step of the said device turning its appropriate instructions into appropriate information that can be conveyed to the said computer via a connection medium includes the step of the said computer listening for a connection on the said connection medium.

56. The method of claim 5 wherein said step of the said computer listening for a connection on the said connection medium includes the step of the said device requesting a connection via the said connection medium to the said computer.

57. The method of claim 56 wherein said step of the said device requesting a connection via the said connection medium to the said computer includes the step of the said computer accepting the said request of the said device for the said connection via the said connection medium

58. The method of claim 57 wherein said step of the said computer accepting the said request of the said device for the said connection via the said connection medium includes the step of the said device transmitting the said appropriate information via the said connection medium to the said computer.

59. The method of claim 58 wherein said step of the said device transmitting the said appropriate information via the said connection medium to the said computer includes

the step of the said computer receiving the said appropriate information from the said device via the said connection medium.

60. The method of claim 59 wherein said step of the said computer receiving the said appropriate information from the said device via the said connection medium includes the step of the said computer examining and storing the said received appropriate information and identifying the said device.

61. The method of claim 51 wherein said step of the said computer creating a complete model being an abstraction of the said device on the said computer includes the step of the said device model abstraction on the said computer being an idea of what it is, its current state and what it can do.

62. The method of claim 11 wherein said step of the said device model abstraction on the said computer being an idea of what it is, its current state and what it can do includes the step of the said device abstraction model on the said computer incorporating all appropriate information received by the said computer from the said device.

63. The method of claim 51 wherein said step of the said computer creating a complete model being an abstraction of the said device on the said computer includes the step of there being an abstraction control simulation of the said device via a bio-directional relationship between the said device and the said device model abstraction on the said computer.

64. The method of claim 51 where said step of the said computer determining and imposing the appropriate behaviour information of the said device model

abstraction on the said computer includes the step of all information and intelligence regarding applications being resident on the said computer.

65. The method of claim 64 wherein said step of all information and intelligence regarding applications being resident on the said computer includes the step of the said computer examining the said device abstraction model on the said computer and comparing it to the applications or information on the said computer to determine the compatibility of the said applications or information with the said device abstraction model.

66. The method of claim 65 wherein said step of the said computer examining the said device abstraction model on the said computer and comparing it to the applications or information on the said computer to determine the compatibility of the said applications or information with the said device abstraction model includes the step of the said computer determining which applications or information on the said computer can be used on the said device abstraction model on the said computer.

67. The method of claim 66 wherein said step of the said computer determining which applications or information on the said computer can be used on the said device abstraction model on the said computer includes the step of the said computer determining how the said device model abstraction on the said computer can use the said applications or information on the said computer.

68. The method of claim 67 wherein said step of the said computer determining how the said device model abstraction on the said computer can use the said applications or information on the said computer includes the step of the said computer changing the said

device model abstraction on the said computer to reflect how the said computer allows the said device model abstraction on the said computer to interact with it.

69. The method of claim 51 wherein said step of the said computer changing the said device model abstraction on the said computer to reflect the said imposed appropriate behaviour information of the said device model abstraction on the said computer includes the step of the said computer identifying the said applications or information that is to be presented on the said device model abstraction on the said computer.

70. The method of claim 69 wherein said step of the said computer identifying the said applications or information that is to be presented on the said device model abstraction on the said computer includes the step of the said computer translating the said applications or information into a format containing the appropriate information that can be presented by the said device model abstraction on the said computer.

71. The method of claim 70 wherein said step of the said computer translating the said applications or information into a format containing the appropriate information that can be presented by the said device model abstraction on the said computer includes the step of the said format containing the appropriate information being composed of instructions for the said device model abstraction on the said computer.

72. The method of claim 71 wherein said step of the said format containing the appropriate information being composed of instructions for the said device model abstraction on the said computer includes the step of the said computer implementing any resulting changes in

the said device model abstraction on the said computer.

73. The method of claim 72 wherein said step of the said computer implementing any resulting changes in the said device model abstraction on the said computer includes the step of the said computer storing the said format containing the said appropriate behaviour information in the said device model abstraction on the said computer.

74. The method of claim 51 wherein said step of the said computer transmitting the said imposed appropriate behaviour information to the said device includes the step of the said computer transmitting the said format containing the said imposed appropriate behaviour information to the said device via the said connection medium.

75. The method of claim 74 wherein said step of the said computer transmitting the said format containing the said imposed appropriate behaviour information to the said device via the said connection medium includes the step of the said device receiving the said format containing the said imposed appropriate behaviour information transmitted by the said computer via the said connection medium.

76. The method of claim 51 wherein said step of the said device presenting the said imposed appropriate behaviour information includes the step of the said device being confined to operating within the said format containing the said imposed appropriate behaviour information transmitted by the said computer.

77. The method of claim 76 wherein the said step of the said device being confined to operating within the said format containing the said imposed appropriate behaviour information transmitted by the said computer includes

the step of the said device behaving in accordance with the said device abstraction model on the said computer.

78. The method of claim 51 wherein said step of the said device presenting the said imposed appropriate behaviour information includes the step of the said device receiving, processing and storing the said format containing the said imposed appropriate behaviour information as projection input.

79. The method of claim 78 wherein said step of the said device receiving, processing and storing the said format containing the said imposed appropriate behaviour information as projection input includes the step of the said device reconstructing the said imposed appropriate behaviour information into the applications, information and instructions contained within the said received format containing the said imposed appropriate behaviour information.

80. The method of claim 79 wherein said step of the said device reconstructing the said imposed appropriate behaviour information into the applications, information and instructions contained within the said received format containing the said imposed appropriate behaviour information includes the step of the said device performing the instructions of the said received format containing the said imposed appropriate behaviour information.

81. The method of claim 80 wherein the said step of the said device performing the instructions of the said received format containing the said imposed appropriate behaviour information includes the step of the said device presenting, displaying or otherwise manifesting the said applications, information and instructions of the said

received format containing the said imposed appropriate behaviour information in such a manner as to generate or otherwise procure an interaction of or with the said device.

82. The method of claim 51 wherein said step of the said device transmitting appropriate response behaviour information to the said computer includes the step of the said device relying on the said format input into the said device and any resulting interaction of or with the said device and the said format input to generate the reaction output being the said appropriate response behaviour information of the said device.

83. The method of claim 82 wherein the said step of the said device relying on the said format input into the said device and any resulting interaction of or with the said device and the said format input to generate the reaction output being the said appropriate response behaviour information of the said device includes the step of the said device recording the said reaction output being the said appropriate response behaviour information on the said device.

84. The method of claim 83 wherein the said step of said device recording the said reaction output being the said appropriate response behaviour information on the said device includes the step of the said device turning device instructions into instructions that can be conveyed to the said computer via the said connection medium.

85. The method of claim 83 wherein the said step of said device turning device instructions into instructions that can be conveyed to the said computer via the said connection medium includes the step of the said device

transmitting the said reaction output being the said appropriate response behaviour information to the said computer via the said communication medium.

86. The method of claim 51 wherein said step of the said device transmitting the said appropriate response behaviour information to the said computer includes the step of the said device being capable or incapable of determining whether any modification has occurred to the said device.

87. The method of claim 86 wherein said step of the said device being capable or incapable of determining whether any modification has occurred to the said device includes the step of the said device being capable or incapable of determining whether any modification has occurred to the said device without relying on the said format input into the said device by the said computer.

88. The method of claim 87 wherein said step of the said device being capable or incapable of determining whether any modification has occurred to the said device without relying on the said format input into the said device by the said computer includes the step of the said device having the said capability determining whether any modification has occurred to the said device without relying on the said format input into the said device by the said computer.

89. The method of claim 88 wherein said step of the said device having the said capability determining whether any modification has occurred to the said device without relying on the said format input into the said device by the said computer includes the step of the said device relying on the said modification occurrence to generate the said reaction output being the said appropriate

response behaviour information of the said device.

90. The method of claim 89 wherein said step of the said device relying on the said modification occurrence to generate the said reaction output being the said appropriate response behaviour information of the said device includes the step of the said device recording the said reaction output being the said appropriate response behaviour information of the said device on the said device.

91. The method of claim 90 wherein said step of the said device recording the said reaction output being the said appropriate response behaviour information of the said device on the said device includes the step of the said method 34 wherein said step of the said device recording the said reaction output being the said appropriate response behaviour information on the said device includes the step of the said device turning device instructions into instructions that can be conveyed to the said computer via the said connection medium.

92. The method of claim 51 wherein said step of the said computer receiving the said appropriate response behaviour information from the said device and transmitting determined and imposed appropriate behaviour information as a response includes the step of the method in claim 57 wherein said step of the said device requesting a connection via the said connection medium to the said computer includes the step of the said computer accepting the said request of the said device for the said connection via the said connection medium.

93. The method of claim 51 wherein said step of the said computer receiving the said appropriate response

behaviour information from the said device and transmitting determined and imposed appropriate behaviour information as a response includes the step of the method of claim 59 wherein the said step of the said device transmitting the said appropriate information via the said connection medium to the said computer includes the said step of the said computer receiving the said appropriate information from the said device via the said connection medium.

94. The method of claim 51 wherein said step of the said computer receiving the said appropriate response behaviour information from the said device and transmitting determined and imposed appropriate behaviour information as a response includes the step of the said device model abstraction being an idea of what it is, its current state and what it can do includes the step of the said device abstraction model incorporating all appropriate information received from the said device.

95. The method of claim 51 wherein said step of the said computer receiving the said appropriate response behaviour information from the said device and transmitting determined and imposed appropriate behaviour information as a response includes the step of the computer determining and imposing the appropriate behaviour information of the said device in abstraction.

96. The method of claim 51 wherein said step of the said computer receiving the said appropriate response behaviour information from the said device and transmitting determined and imposed appropriate behaviour information as a response includes the method of claim 1 wherein said step of the said computer

changing the said model abstraction of the said device to reflect the imposed appropriate behaviour information of the said device.

97. The method of claim 51 wherein said step of the said computer receiving the said appropriate response behaviour information from the said device and transmitting determined and imposed appropriate behaviour information as a response includes the said step of the said computer transmitting the said appropriate behaviour information to the said device.

98. A method of implementing a client user interface for an input/output device which is adapted to be interconnected with a software application, the method comprising the steps of:

defining a first series of classes that simulate a model of the functional capabilities of the input/output device;

defining a second series of classes corresponding to the programs running on said device;

wherein to perform an action, said software application interacts with said first series of classes and upon an interaction, said first series of classes sends output command events corresponding to said action to said second series of classes running on said device; and

said second series of classes sends input command events to said first series of classes which translates said input command events into corresponding input actions and notifies said software application of said input action.

99. A method as claimed in claim 98 wherein said

input/output device comprises one of a palm computer, a mobile phone, a handheld computer or a desktop computer.

100. A method of implementing an interaction between an input/output device and at least one software application running on a server computer, the method comprising the steps of:

interconnecting said input/output device with said software application;

said input/output device and said software application negotiating a set of input/output capabilities for utilization in interacting with said input/output device;

said software application constructing a series of objects corresponding to said set of input/output capabilities and interacting with said series of objects with the interaction resulting in the objects sending output command events corresponding to said interaction to said input/output device; and

said input/output device sending input command events to said series of objects which translates said input command events into corresponding input actions and notifies said software application of said input action.

101. A method as claimed in claim 100 wherein said input/output device interacts with multiple software applications.

102. A system when implementing the method of any one of claims 51 to 101.

103. A device which includes a client capability set definition library; said device adapted to be in communication with a remote computer.

104. A remote computer arranged to execute a control

application which makes reference to a capability set definition library or model.

105. Media incorporating software which implements the system of any one of claims 1 to 50.

106. Media incorporating software which implements the method of any one of claims 51 to 101.

AMENDED CLAIMS
{received by the International Bureau on 8 January 2001 (08.01.01); original claims 26-106 replaced by new claims 26-116; other claims unchanged (22 pages)}

of operation of said device are determined by said control application with reference to said capability set definition library.

28. A distributed client/server system for at least a first device and

a first remote computer

said first device arranged to execute a client application which makes reference to an at least first client capability set driver;

said at least first remote computer arranged to execute a control application which makes reference to an at least first capability set definition library;

said system operating sequentially in the following steps:

(a) said client application on said device initiates communication with said control application on said remote computer for the purpose of establishing a communications session;

(b) said client application and said control application agree on a capability set definition library and a corresponding client capability set driver for use during said communication session;

(c) said control application and said client application maintaining communication during said communication session whereby predetermined aspects of operation of said device are determined by said control application with reference to said selected capability set definition library and said corresponding selected client capability set driver.

29. The system of Claim 27 wherein said capability set definition library contains a set of definitions of

predetermined, selected capabilities of said device.

30. The system of Claim 29 wherein said predetermined capabilities are primitive functions of the device.

31. The system of Claim 29 wherein said capabilities are selected in order to minimise traffic during said communications session.

32. The system of claim 27 wherein said communications session includes the communicating of update information pertinent to the status of individual ones of said capabilities invoked with the aid of said capability set definition library.

33. The system of claim 32 wherein said communications session occurs over a network link.

34. A distributed system for computer interaction; said system including

a device having computing means therein and

a remote computer located or locatable remotely from said device, communication means whereby bi-directional communication is established between said device and said remote computer;

said device arranged to execute a client application; said remote computer arranged to execute a control application.

35. The system of Claim 34 wherein said device makes reference to a client capability set definition library located on said device.

36. The system of Claim 35 wherein said control application makes reference to said capability set definition library.

37. The system of Claim 35 wherein said capability set definition library resides on both said device and said remote computer.

38. The system of Claim 34 wherein said system operates sequentially in the following steps:

(a) said client application on said device initiates communication with said control application on said remote computer for the purpose of establishing a communications session between said device and said remote computer;

(b) said client application and said control application agree on mutual use of said capability set definition library;

(c) said control application and said client application maintaining communication during said communication session whereby predetermined aspects of operation of said device are determined by said control application with reference to said capability set definition library.

39. The system of Claim 35 or 36 wherein said capability set definition library contains a set of definitions of predetermined selected capabilities of said device.

40. The system of Claim 39 wherein said predetermined capabilities are primitive functions of the device.

41. The system of Claim 39 wherein said capabilities are selected in order to minimise traffic during said communication session.

42. The system of claim 39 wherein said communication session includes the communicating of update information pertinent to the status of individual ones of the capabilities invoked with the aid of said capability set definition library.

43. In a distributed system for computer interaction; said system including

a device having computing means therein and

a remote computer located or locatable remotely from said device, said device arranged to execute a client application; said remote computer arranged to execute a control application; a method of operating said system according to the following steps:

(a) making available a capability set definition library on both said device and said remote computer;

(b) said client application on said device initiates communication with said control application on said remote computer for the purpose of establishing a communications session between said device and said remote computer;

(c) said client application and said control application agree on mutual use of said capability set definition library;

(d) said control application and said client application maintaining communication during said communication session whereby predetermined aspects of operation of said, device are determined by said control application with reference to said capability set definition library.

44. The system of Claim 34 wherein said remote computer builds a model of the state of said client application so as to interact with said device by interacting with said model.

45. The system of Claim 44 wherein the said model is continuously updated so as to reflect current status of said client application on said device.

46. The system of Claim 45 wherein said device is arranged to execute said client application with reference to a client capability set definition library located on said

device and said remotc computer is arranged to execute said control application by making reference to a corresponding capability set definition library located on said remote computer.

47. The system of Claim 46 wherein said capability set definition library comprises a plurality of capability definitions, each of said definitions determined with reference to the desired functionality of said device.

48. The system of Claim 47 wherein each of said capability definitions comprises a primitive function of said device.

49. The system of Claim 47 wherein each of said capability definitions comprises a combination of primitive functions of said device.

50. A distributed client/server system for at least a first device and

a first remote computer;

communication means whereby bi-directional communication is established between said device and said remote computer;

said first device arranged to execute a client application which makes reference to an at least first client capability set driver;

said at least first remote computer arranged to execute a control application which makes reference to an at least first capability set definition library;

said system operating sequentially in the following steps:

(a) said client application on said device initiates communication with said control application on said remote computer for the purpose of establishing a communications session;

(b) said client application and said control application agree on a capability set definition library and a corresponding client capability set driver for use during said communication session;

(c) said control application and said client application maintaining communication during said communication session whereby predetermined aspects of operation of said device are determined by said control application with reference to said selected capability set definition library and said corresponding selected client capability set driver;

said system arranged to communicate codes over said communication means which make reference to capability definitions in said capability set definition library thereby to control the volume of traffic on said communication means.

51. A method and system for facilitating a computer controlling an interaction with a device using a model of the device:

- The said device asking for a connection to the said computer
- The said computer creating a complete model being an abstraction of the said device on the said computer
- The said computer determining and imposing the appropriate behaviour information of the said device model abstraction on the said computer
- The said computer changing the said device model abstraction on the said computer to reflect the said imposed appropriate behaviour information of

the said device model abstraction on the said computer

- The said computer transmitting the said imposed appropriate behaviour information to the said device
- The said device presenting the said imposed appropriate behaviour information
- The said device transmitting appropriate response behaviour information to the said computer
- The said computer receiving the said appropriate response behaviour information from the said device and transmitting determined and imposed appropriate behaviour information as a response.

52. The method of claim 51 wherein said step of the said device asking for a connection to the said computer includes the step of the said device determining appropriate instructions about what it is, its current state and what it can do.

53. The method of claim 52 wherein said step of the said device determining appropriate instructions about what it is, its current state and what it can do includes the step of the said device determining what applications or information contained on the said computer the said device requires.

54. The method of claim 53 wherein said step of the said device determining what applications or information contained on the said computer the said device requires includes the step the said device turning its appropriate instructions into appropriate information that can be conveyed to the said computer via a connection medium.

55. The method of claim 54 wherein said step of the said device turning its appropriate instructions into appropriate information that can be conveyed to the said computer via a connection medium includes the step of the said computer listening for a connection on the said connection medium.

56. The method of claim 5 wherein said step of the said computer listening for a connection on the said connection medium includes the step of the said device requesting a connection via the said connection medium to the said computer.

57. The method of claim 56 wherein said step of the said device requesting a connection via the said connection medium to the said computer includes the step of the said computer accepting the said request of the said device for the said connection via the said connection medium

58. The method of claim 57 wherein said step of the said computer accepting the said request of the said device for the said connection via the said connection medium includes the step of the said device transmitting the said appropriate information via the said connection medium to the said computer.

59. The method of claim 58 wherein said step of the said device transmitting the said appropriate information via the said connection medium to the said computer includes the step of the said computer receiving the said appropriate information from the said device via the said connection medium.

60. The method of claim 59 wherein said step of the said computer receiving the said appropriate information from the said device via the said connection medium includes

the step of the said computer examining and storing the said received appropriate information and identifying the said device.

61. The method of claim 51 wherein said step of the said computer creating a complete model being an abstraction of the said device on the said computer includes the step of the said device model abstraction on the said computer being an idea of what it is, its current state and what it can do.

62. The method of claim 51 wherein said step of the said device model abstraction on the said computer being an idea of what it is, its current state and what it can do includes the step of the said device abstraction model on the said computer incorporating all appropriate information received by the said computer from the said device.

63. The method of claim 51 wherein said step of the said computer creating a complete model being an abstraction of the said device on the said computer includes the step of there being an abstraction control simulation of the said device via a bi-directional relationship between the said device and the said device model abstraction on the said computer.

64. The method of claim 51 where said step of the said computer determining and imposing the appropriate behaviour information of the said device model abstraction on the said computer includes the step of all information and intelligence regarding applications being resident on the said computer.

65. The method of claim 64 wherein said step of all information and intelligence regarding applications being resident on the said computer includes the step of

the said computer examining the said device abstraction model on the said computer and comparing it to the applications or information on the said computer to determine the compatibility of the said applications or information with the said device abstraction model.

66. The method of claim 65 wherein said step of the said computer examining the said device abstraction model on the said computer and comparing it to the applications or information on the said computer to determine the compatibility of the said applications or information with the said device abstraction model includes the step of the said computer determining which applications or information on the said computer can be used on the said device abstraction model on the said computer.

67. The method of claim 66 wherein said step of the said computer determining which applications or information on the said computer can be used on the said device abstraction model on the said computer includes the step of the said computer determining how the said device model abstraction on the said computer can use the said applications or information on the said computer.

68. The method of claim 67 wherein said step of the said computer determining how the said device model abstraction on the said computer can use the said applications or information on the said computer includes the step of the said computer changing the said device model abstraction on the said computer to reflect how the said computer allows the said device model abstraction on the said computer to interact with it.

69. The method of claim 51 wherein said step of the said computer changing the said device model abstraction on the said computer to reflect the said imposed

appropriate behaviour information of the said device model abstraction on the said computer includes the step of the said computer identifying the said applications or information that is to be presented on the said device model abstraction on the said computer.

70. The method of claim 69 wherein said step of the said computer identifying the said applications or information that is to be presented on the said device model abstraction on the said computer includes the step of the said computer translating the said applications or information into a format containing the appropriate information that can be presented by the said device model abstraction on the said computer.

71. The method of claim 70 wherein said step of the said computer translating the said applications or information into a format containing the appropriate information that can be presented by the said device model abstraction on the said computer includes the step of the said format containing the appropriate information being composed of instructions for the said device model abstraction on the said computer.

72. The method of claim 71 wherein said step of the said format containing the appropriate information being composed of instructions for the said device model abstraction on the said computer includes the step of the said computer implementing any resulting changes in the said device model abstraction on the said computer.

73. The method of claim 72 wherein said step of the said computer implementing any resulting changes in the said device model abstraction on the said computer includes the step of the said computer storing the said format containing the said appropriate behaviour information in

the said device model abstraction on the said computer.

74. The method of claim 51 wherein said step of the said computer transmitting the said imposed appropriate behaviour information to the said device includes the step of the said computer transmitting the said format containing the said imposed appropriate behaviour information to the said device via the said connection medium.

75. The method of claim 74 wherein said step of the said computer transmitting the said format containing the said imposed appropriate behaviour information to the said device via the said connection medium includes the step of the said device receiving the said format containing the said imposed appropriate behaviour information transmitted by the said computer via the said connection medium.

76. The method of claim 51 wherein said step of the said device presenting the said imposed appropriate behaviour information includes the step of the said device being confined to operating within the said format containing the said imposed appropriate behaviour information transmitted by the said computer.

77. The method of claim 76 wherein the said step of the said device being confined to operating within the said format containing the said imposed appropriate behaviour information transmitted by the said computer includes the step of the said device behaving in accordance with the said device abstraction model on the said computer.

78. The method of claim 51 wherein said step of the said device presenting the said imposed appropriate behaviour information includes the step of the said device receiving, processing and storing the said format

containing the said imposed appropriate behaviour information as projection input.

79. The method of claim 78 wherein said step of the said device receiving, processing and storing the said format containing the said imposed appropriate behaviour information as projection input includes the step of the said device reconstructing the said imposed appropriate behaviour information into the applications, information and instructions contained within the said received format containing the said imposed appropriate behaviour information.

80. The method of claim 79 wherein said step of the said device reconstructing the said imposed appropriate behaviour information into the applications, information and instructions contained within the said received format containing the said imposed appropriate behaviour information includes the step of the said device performing the instructions of the said received format containing the said imposed appropriate behaviour information.

81. The method of claim 80 wherein the said step of the said device performing the instructions of the said received format containing the said imposed appropriate behaviour information includes the step of the said device presenting, displaying or otherwise manifesting the said applications, information and instructions of the said received format containing the said imposed appropriate behaviour information in such a manner as to generate or otherwise procure an interaction of or with the said device.

82. The method of claim 51 wherein said step of the said device transmitting appropriate response behaviour

information to the said computer includes the step of the said device relying on the said format input into the said device and any resulting interaction of or with the said device and the said format input to generate the reaction output being the said appropriate response behaviour information of the said device.

83. The method of claim 82 wherein the said step of the said device relying on the said format input into the said device and any resulting interaction of or with the said device and the said format input to generate the reaction output being the said appropriate response behaviour information of the said device includes the step of the said device recording the said reaction output being the said appropriate response behaviour information on the said device.

84. The method of claim 83 wherein the said step of said device recording the said reaction output being the said appropriate response behaviour information on the said device includes the step of the said device turning device instructions into instructions that can be conveyed to the said computer via the said connection medium.

85. The method of claim 83 wherein the said step of said device turning device instructions into instructions that can be conveyed to the said computer via the said connection medium includes the step of the said device transmitting the said reaction output being the said appropriate response behaviour information to the said computer via the said communication medium.

86. The method of claim 51 wherein said step of the said device transmitting the said appropriate response behaviour information to the said computer includes the

step of the said device being capable or incapable of determining whether any modification has occurred to the said device.

87. The method of claim 86 wherein said step of the said device being capable or incapable of determining whether any modification has occurred to the said device includes the step of the said device being capable or incapable of determining whether any modification has occurred to the said device without relying on the said format input into the said device by the said computer.

88. The method of claim 87 wherein said step of the said device being capable or incapable of determining whether any modification has occurred to the said device without relying on the said format input into the said device by the said computer includes the step of the said device having the said capability determining whether any modification has occurred to the said device without relying on the said format input into the said device by the said computer.

89. The method of claim 88 wherein said step of the said device having the said capability determining whether any modification has occurred to the said device without relying on the said format input into the said device by the said computer includes the step of the said device relying on the said modification occurrence to generate the said reaction output being the said appropriate response behaviour information of the said device.

90. The method of claim 89 wherein said step of the said device relying on the said modification occurrence to generate the said reaction output being the said appropriate response behaviour information of the said device includes the step of the said device recording

the said reaction output being the said appropriate response behaviour information of the said device on the said device.

91. The method of claim 90 wherein said step of the said device recording the said reaction output being the said appropriate response behaviour information of the said device on the said device includes the step of the said method 34 wherein said step of the said device recording the said reaction output being the said appropriate response behaviour information on the said device includes the step of the said device turning device instructions into instructions that can be conveyed to the said computer via the said connection medium.

92. The method of claim 51 wherein said step of the said computer receiving the said appropriate response behaviour information from the said device and transmitting determined and imposed appropriate behaviour information as a response includes the step of the method in claim 57 wherein said step of the said device requesting a connection via the said connection medium to the said computer includes the step of the said computer accepting the said request of the said device for the said connection via the said connection medium.

93. The method of claim 51 wherein said step of the said computer receiving the said appropriate response behaviour information from the said device and transmitting determined and imposed appropriate behaviour information as a response includes the step of the method of claim 59 wherein the said step of the said device transmitting the said appropriate information via the said connection medium to the said computer includes

the said step of the said computer receiving the said appropriate information from the said device via the said connection medium.

94. The method of claim 51 wherein said step of the said computer receiving the said appropriate response behaviour information from the said device and transmitting determined and imposed appropriate behaviour information as a response includes the step of the said device model abstraction being an idea of what it is, its current state and what it can do includes the step of the said device abstraction model incorporating all appropriate information received from the said device.

95. The method of claim 51 wherein said step of the said computer receiving the said appropriate response behaviour information from the said device and transmitting determined and imposed appropriate behaviour information as a response includes the step of the computer determining and imposing the appropriate behaviour information of the said device in abstraction.

96. The method of claim 51 wherein said step of the said computer receiving the said appropriate response behaviour information from the said device and transmitting determined and imposed appropriate behaviour information as a response includes the step of the said computer changing the said model abstraction of the said device to reflect the imposed appropriate behaviour information of the said device.

97. The method of claim 51 wherein said step of the said computer receiving the said appropriate response behaviour information from the said device and transmitting determined and imposed appropriate

behaviour information as a response includes the said step of the said computer transmitting the said appropriate behaviour information to the said device.

98. A method of implementing a client user interface for an input/output device which is adapted to be interconnected with a software application, the method comprising the steps of:

defining a first series of classes that simulate a model of the functional capabilities of the input/output device;

defining a second series of classes corresponding to the programs running on said device;

wherein to perform an action, said software application interacts with said first series of classes and upon an interaction, said first series of classes sends output command events corresponding to said action to said second series of classes running on said device; and

said second series of classes sends input command events to said first series of classes which translates said input command events into corresponding input actions and notifies said software application of said input action.

99. A method as claimed in claim 98 wherein said input/output device comprises one of a palm computer, a mobile phone, a handheld computer or a desktop computer.

100. A method of implementing an interaction between an input/output device and at least one software application running on a server computer, the method comprising the steps of:

interconnecting said input/output device with said

software application;

said input/output device and said software application negotiating a set of input/output capabilities for utilization in interacting with said input/output device;

said software application constructing a series of objects corresponding to said set of input/output capabilities and interacting with said series of objects with the interaction resulting in the objects sending output command events corresponding to said interaction to said input/output device; and

said input/output device sending input command events to said series of objects which translates said input command events into corresponding input actions and notifies said software application of said input action.

101. A method as claimed in claim 100 wherein said input/output device interacts with multiple software applications.

102. A system when implementing the method of any one of claims 51 to 101.

103. A device which includes a client capability set definition library; said device adapted to be in communication with a remote computer.

104. A remote computer arranged to execute a control application which makes reference to a capability set definition library.

105. Media incorporating software which implements the system of any one of claims 1 to 50.

106. Media incorporating software which implements the method of any one of claims 51 to 101.

107. A remote computer arranged to execute a control

application which makes reference to a device model abstraction.

108. A distributed system for computer interaction; said system including

a device having computing means therein and

a remote computer located or locatable remotely from said device, said device arranged to execute a client application which makes reference to a client capability set definition library located on said device; said remote computer arranged to execute a control application which makes reference to a corresponding capability set definition library located on said remote computer.

109. The system of Claim 108 further including communication means whereby bi-directional communication is established between said device and said remote computer.

110. The system of Claim 108 or 109 wherein said client application and said control application agree on mutual use of said capability set definition library.

111. The system of any one of Claims 108, 109 or 110 wherein said control application and said client application maintain communication during said communication session whereby predetermined aspects of operation of said device are determined by said control application with reference to said capability set definition library for the purpose of minimizing or controlling data flow between said device and said remote computer.

112. The system of any one of Claims 108 to 111 wherein said system operates sequentially in the following steps:

(a) said client application on said device initiates communication with said control application on said

remote computer for the purpose of establishing a communications session between said device and said remote computer;

(b) said client application and said control application agree on mutual use of said capability set definition library;

(c) said control application and said client application maintaining communication during said communication session whereby predetermined aspects of operation of said device are determined by said control application with reference to said capability set definition library.

113. A computer readable medium containing code which implements the system of any one of Claims 1 to 17 or the system of any one of Claims 19 to 50.

114. A computer readable medium containing code which permits a computer to implement the method of any one of Claims 18 or 51 to 101.

115. A computer readable medium containing code which permits the device of Claim 103 to implement said client capability set definition library.

116. A computer readable medium containing code which permits the remote computer of Claim 104 to execute said control application.

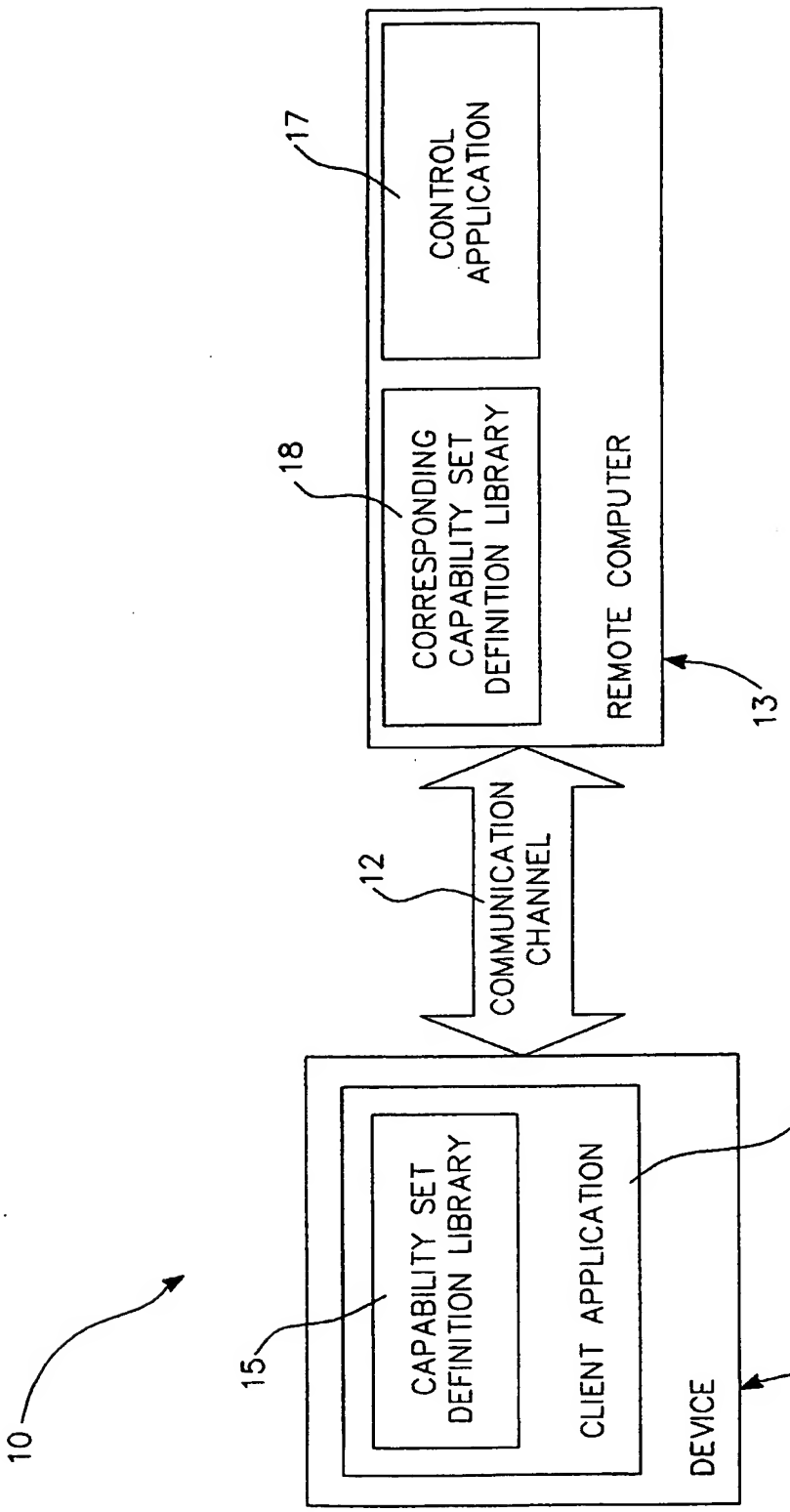

10
11
12
13
14
15
17
18
CONTROL APPLICATION
CORRESPONDING CAPABILITY SET DEFINITION LIBRARY
REMOTE COMPUTER
COMMUNICATION CHANNEL
CAPABILITY SET DEFINITION LIBRARY
CLIENT APPLICATION
DEVICE

Fig. 1

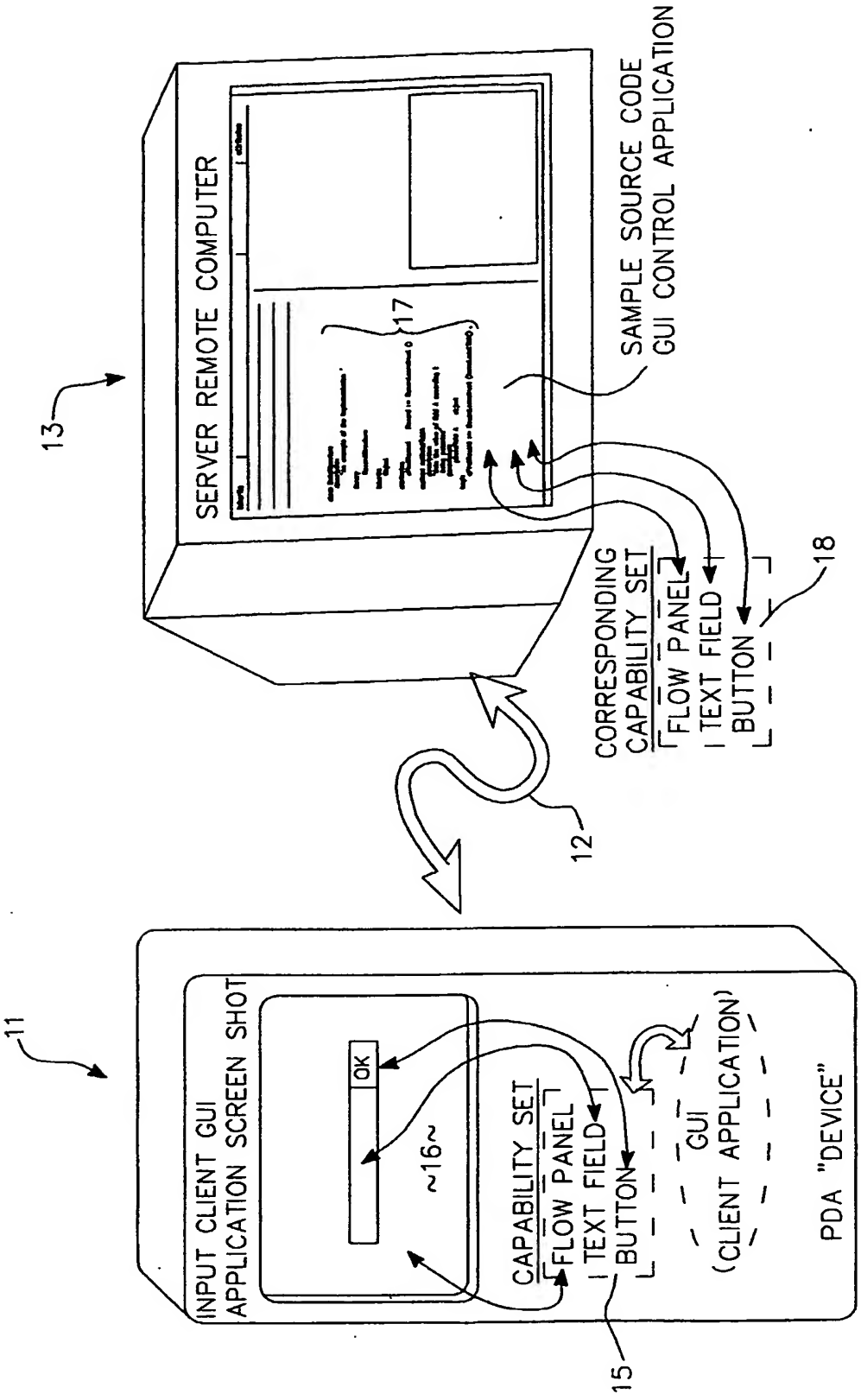

SERVER REMOTE COMPUTER
SAMPLE SOURCE CODE
GUI CONTROL APPLICATION
CORRESPONDING
CAPABILITY SET
FLOW PANEL
TEXT FIELD
BUTTON
INPUT CLIENT GUI
APPLICATION SCREEN SHOT
OK
CAPABILITY SET
FLOW PANEL
TEXT FIELD
BUTTON
GUI
(CLIENT APPLICATION)
PDA "DEVICE"
11
12
13
15
16
17
18

Fig. 2

Sample Source Code - GUI 17 Control Application
File: D:\work\Engineering-Processes\Patents\Antenna\input\input.ant 18/12/99. 16:48:53

```
1   class InputApplication
2
3   description
4   "A bullant Remote demonstration program"
5
6   library
7     InputDemo
8
9   author
10    "Bullant Technology Pty Ltd"
11
12  inherits
13  DesktopGUIApplication, ButtonWatcher,
    Command
14
15  attributes
16    aFrame  Frame
17    aTextField        TextField
18
19  #---------------------------------------#
20  method start
21    description
22      "Start the input demonstration"
23    exposure
24      public
25    returns
26      rResult        Object
27  begin
28    registerApplication ("input")
29    return null
30  end method
31
32  #---------------------------------------#
33  method newConnection
34    parameters
35      pConnection    RDPConnection,
36      pProperties    Properties
37  locals
38  lFlowPanel FlowPanel :=Construct(FlowPanel)
39  lButton   Button :=Construct (Button, "OK")
40  begin
41   aFrame :=Construct (Frame, "Test frame",
    PConnection)
42    aTextfield :- Construct (TextField)
43    lFlowPanel.add(aTextField)
44    lButton.registerWatcher(self)
45    lFlowPanel.add(lButton)
46    aFrame.setContent (lFlowPanel)
47    aFrame.setVisible(true)
48  end method
49
50  #-------------------------------------#
51  method buttonClick
52    parameters
53      pButton        Button
54    locals
55      lDialogDialog:=Construct(Dialog.
    AFrame. ATextField.getText())
56  begin
57    lDialog.setVisible(true)
58    end method
59  end class
```

Fig. 3

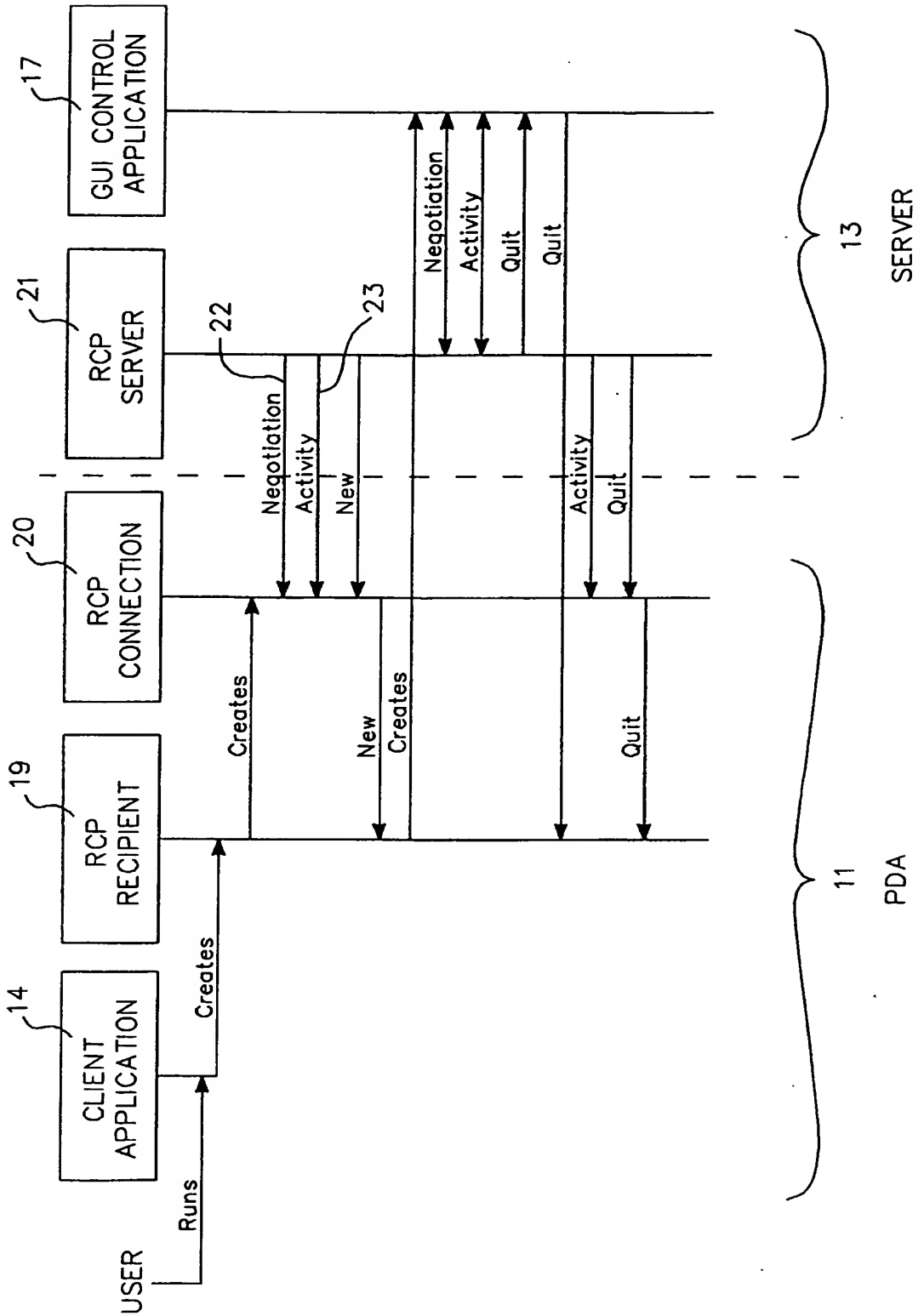

USER
Runs
14
CLIENT APPLICATION
Creates
19
RCP RECIPIENT
Creates
New
Creates
Quit
20
RCP CONNECTION
Negotiation
Activity
New
Activity
Quit
21
RCP SERVER
22
23
Negotiation
Activity
Quit
Quit
17
GUI CONTROL APPLICATION
11
PDA
13
SERVER

Fig. 4

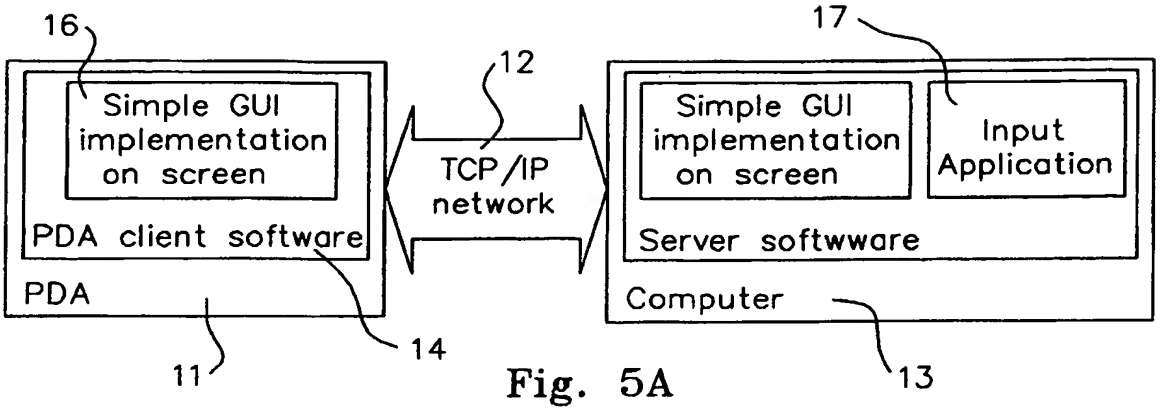


Fig. 5A

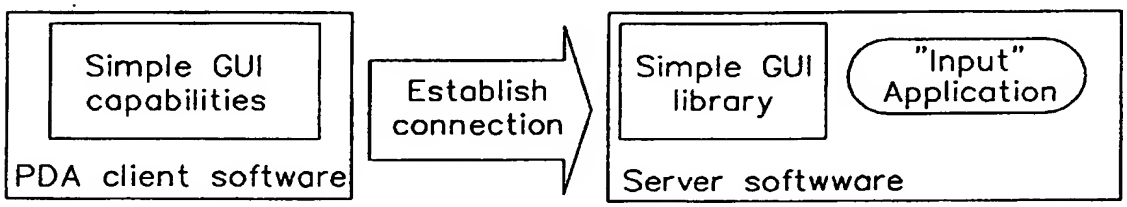


Fig. 5B

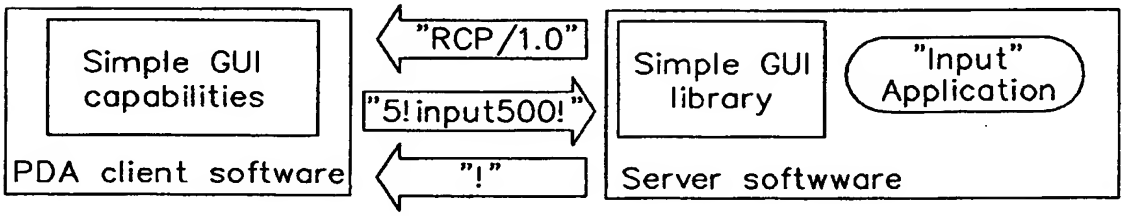


Fig. 5C

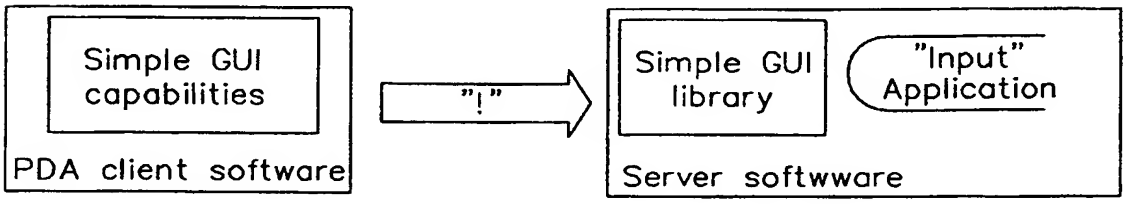


Fig. 5D

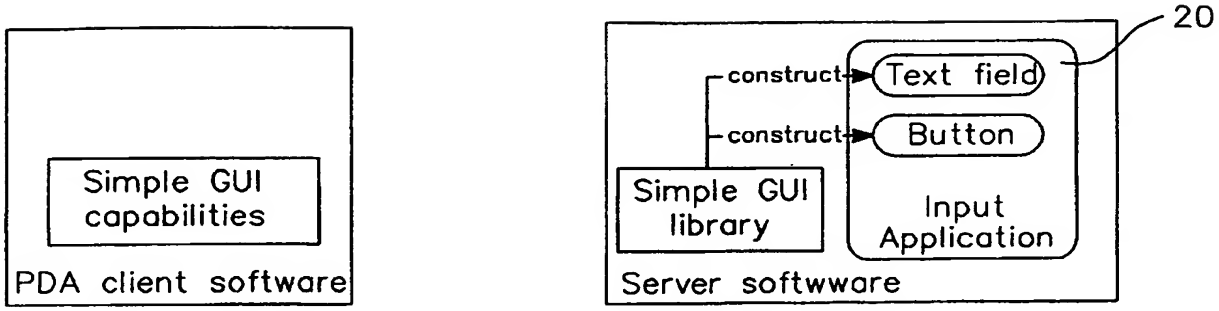


Fig. 5E

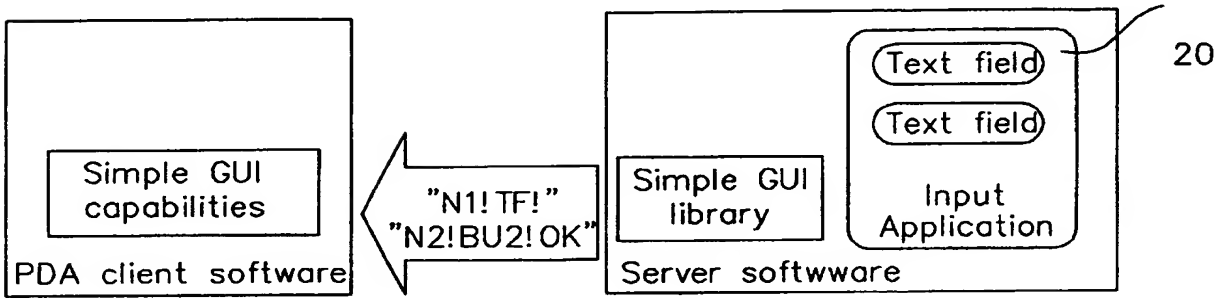


Fig. 5F

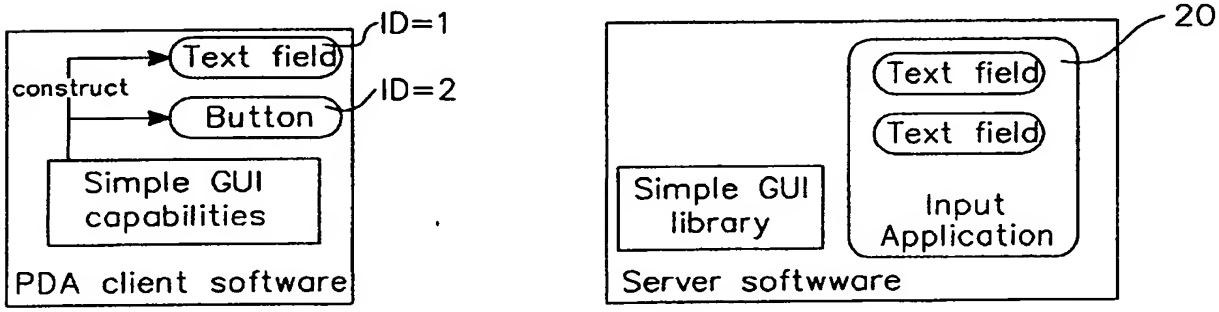


Fig. 5G

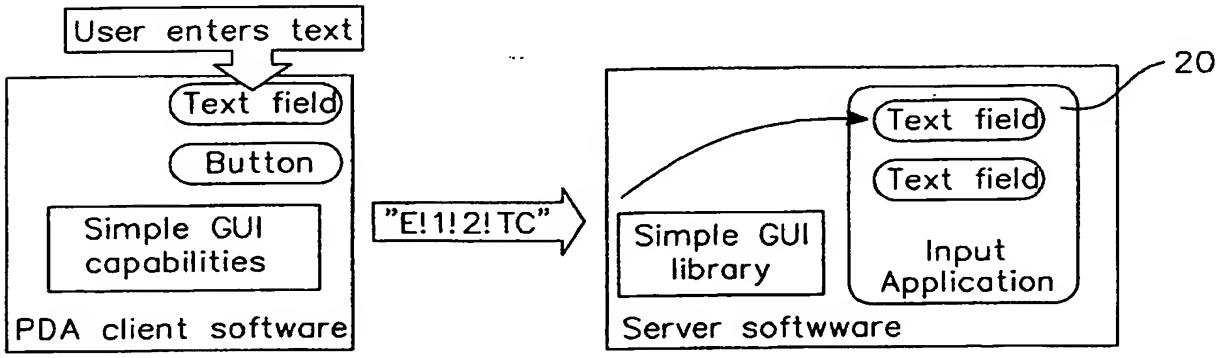


Fig. 5H

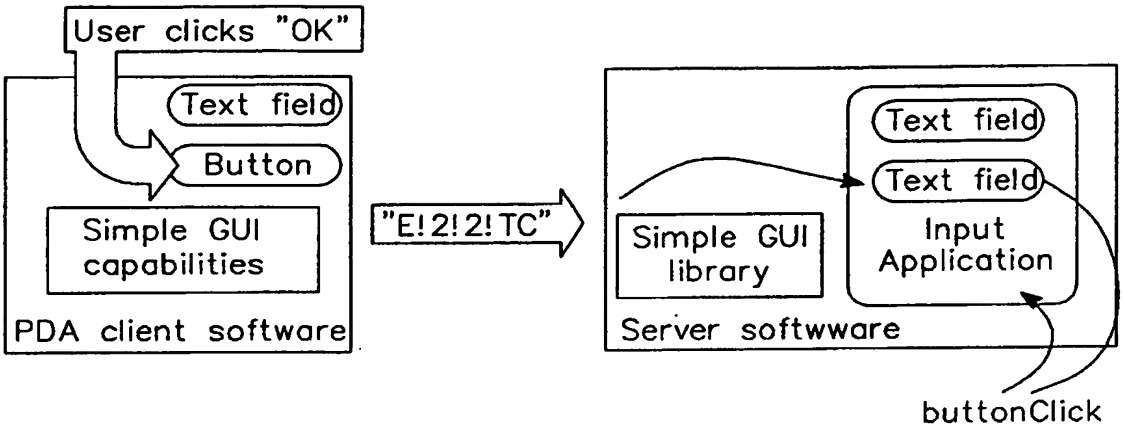

User clicks "OK"
Text field
Button
Simple GUI capabilities
PDA client software
"E!2!2!TC"
Text field
Text field
Simple GUI library
Input Application
Server softwware
buttonClick

Fig. 5I

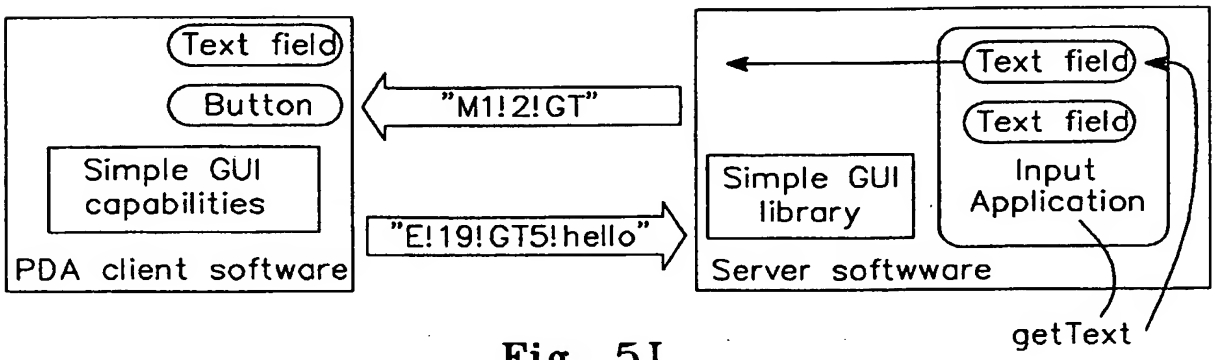

Text field
Button
Simple GUI capabilities
PDA client software
"M1!2!GT"
"E!19!GT5!hello"
Text field
Text field
Simple GUI library
Input Application
Server softwware
getText

Fig. 5J

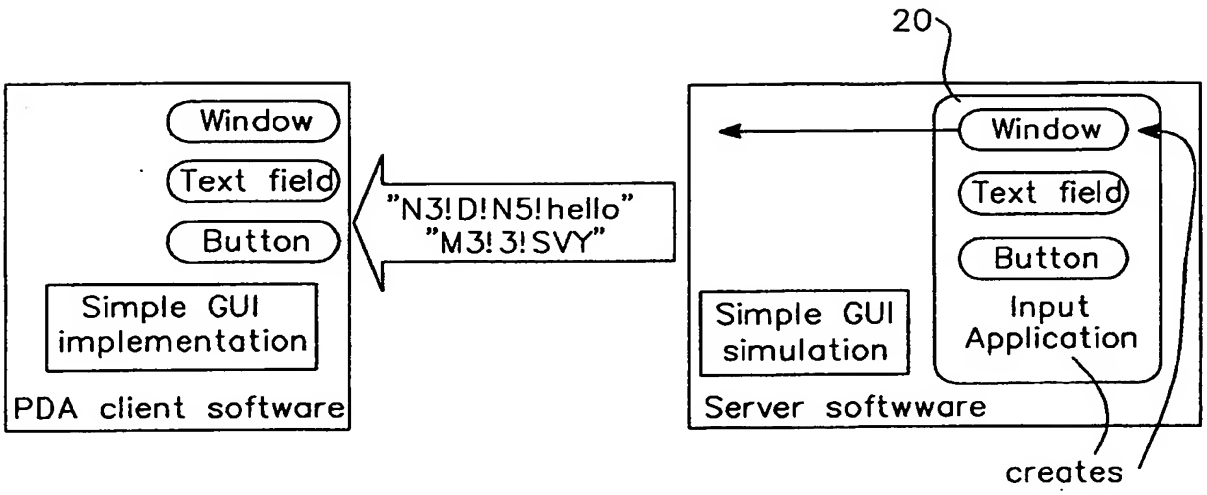

20
Window
Text field
Button
Simple GUI implementation
PDA client software
"N3!D!N5!hello"
"M3!3!SVY"
Window
Text field
Button
Simple GUI simulation
Input Application
Server softwware
creates

Fig. 5K

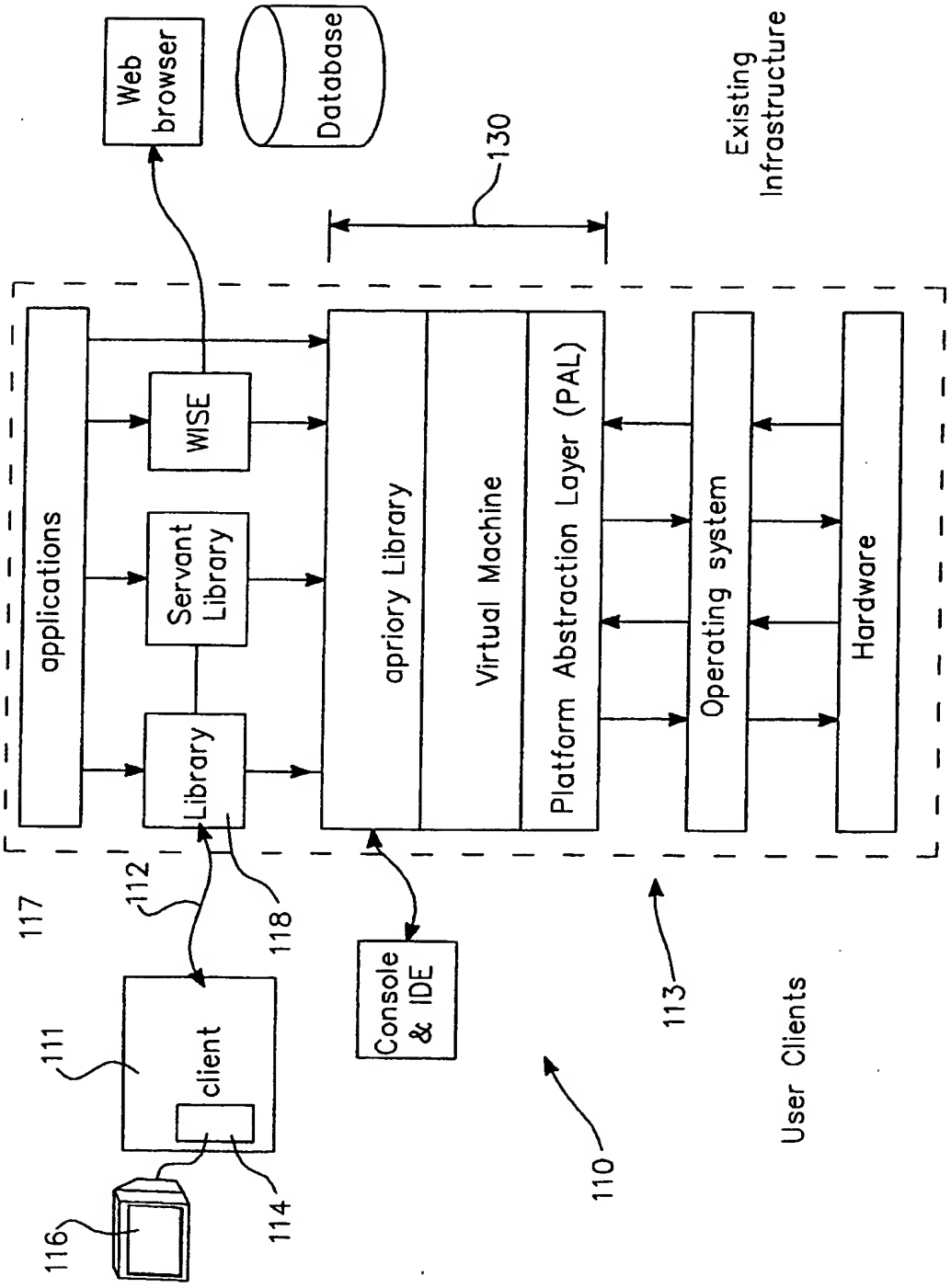

Web browser
Database
130
Existing Infrastructure
applications
WISE
Servant Library
Library
apriory Library
Virtual Machine
Platform Abstraction Layer (PAL)
Operating system
Hardware
117
112
118
Console & IDE
113
User Clients
111
client
116
114
110

Fig. 6

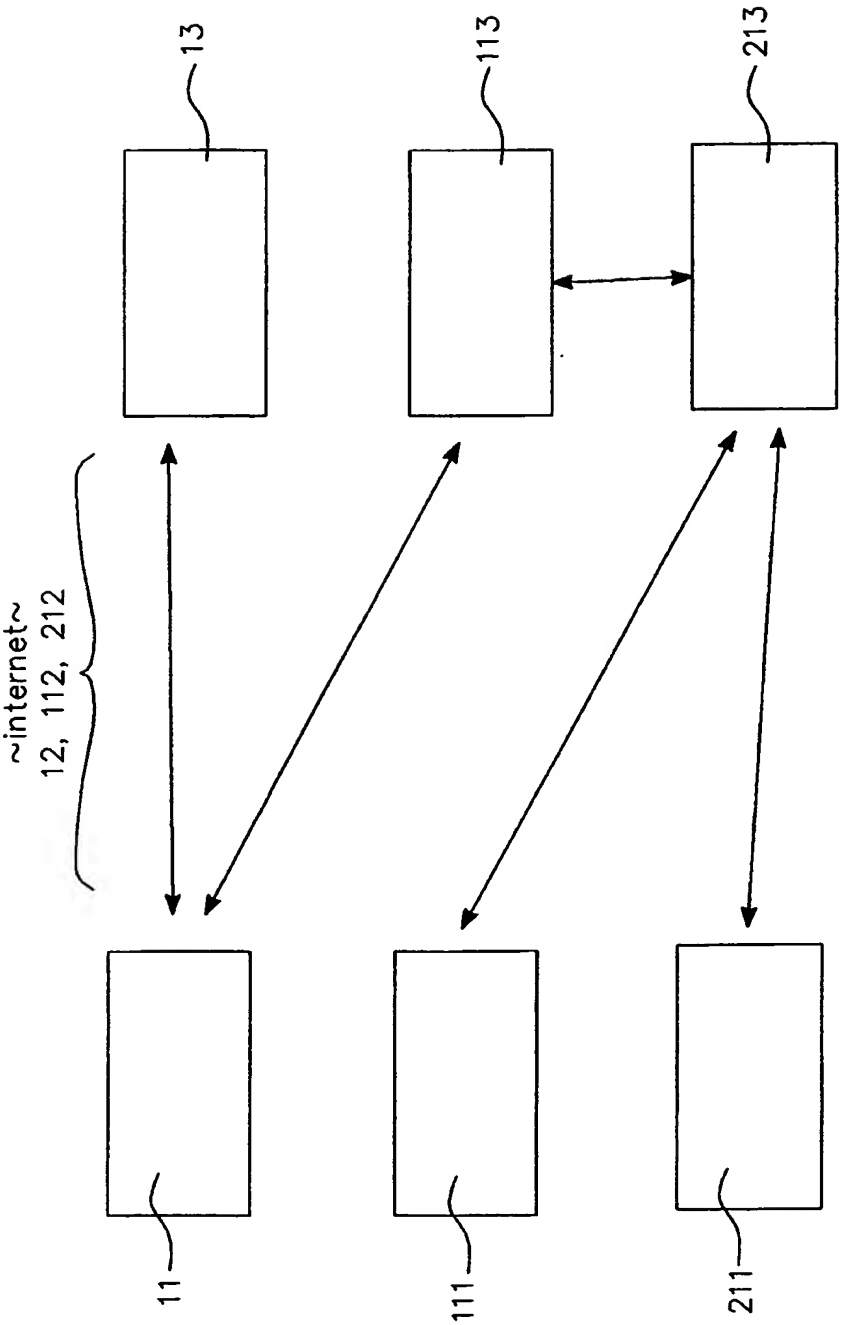


Fig. 7

CAPABILITIES

- receive call
- initiate call
- display character
- voice recognize word

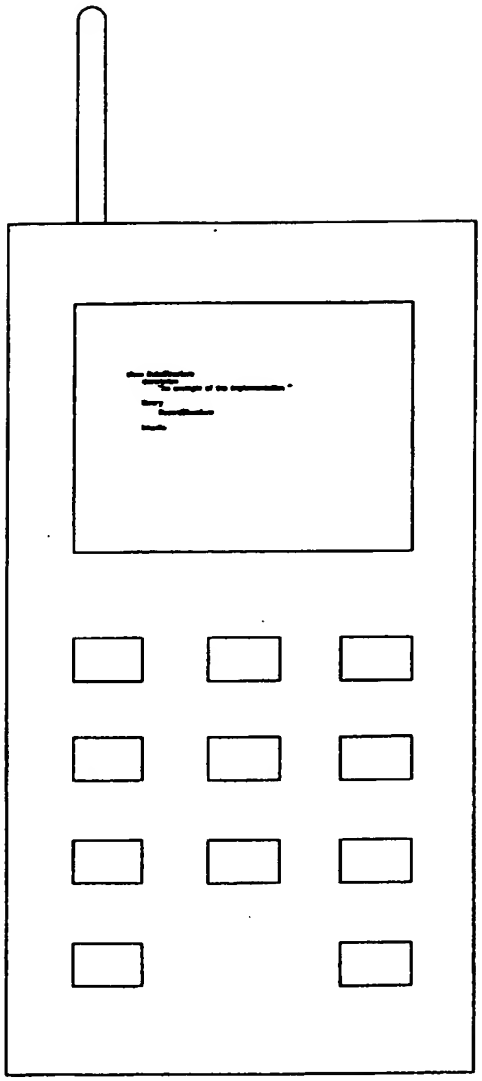


Fig. 8

CAPABILITIES

- go to floor
- open door
- close door

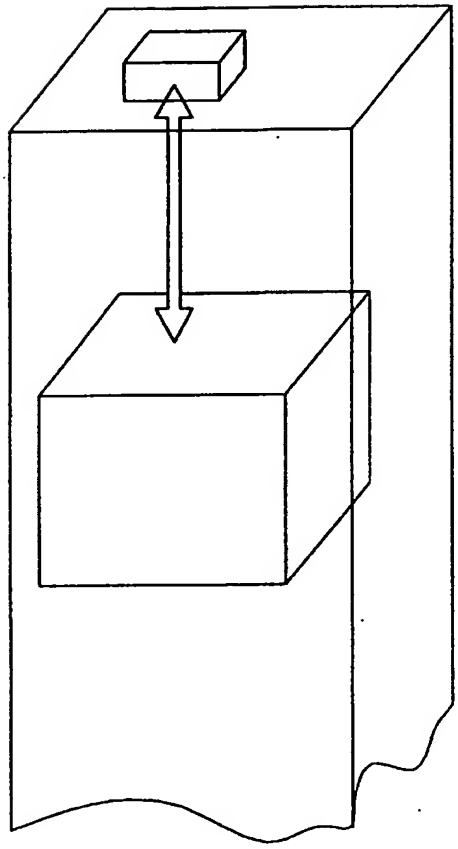


Fig. 9

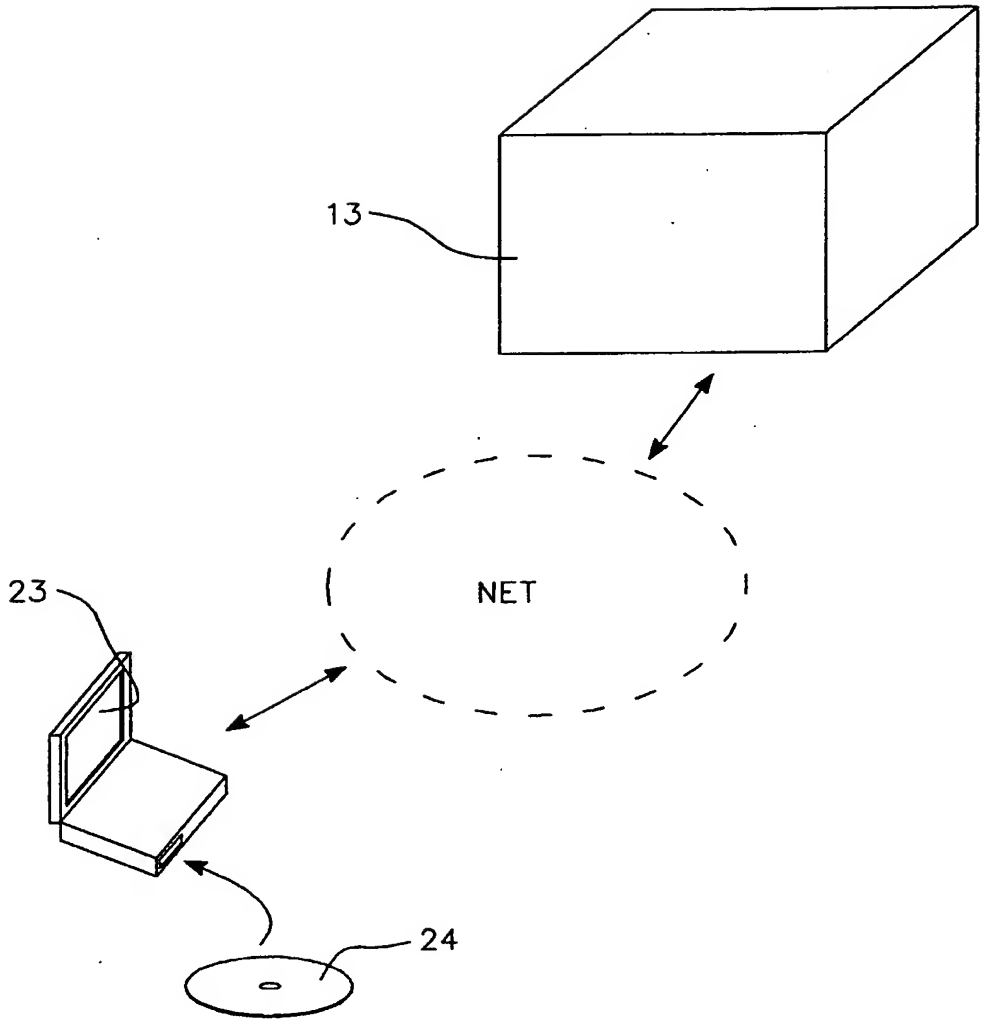


Fig. 10

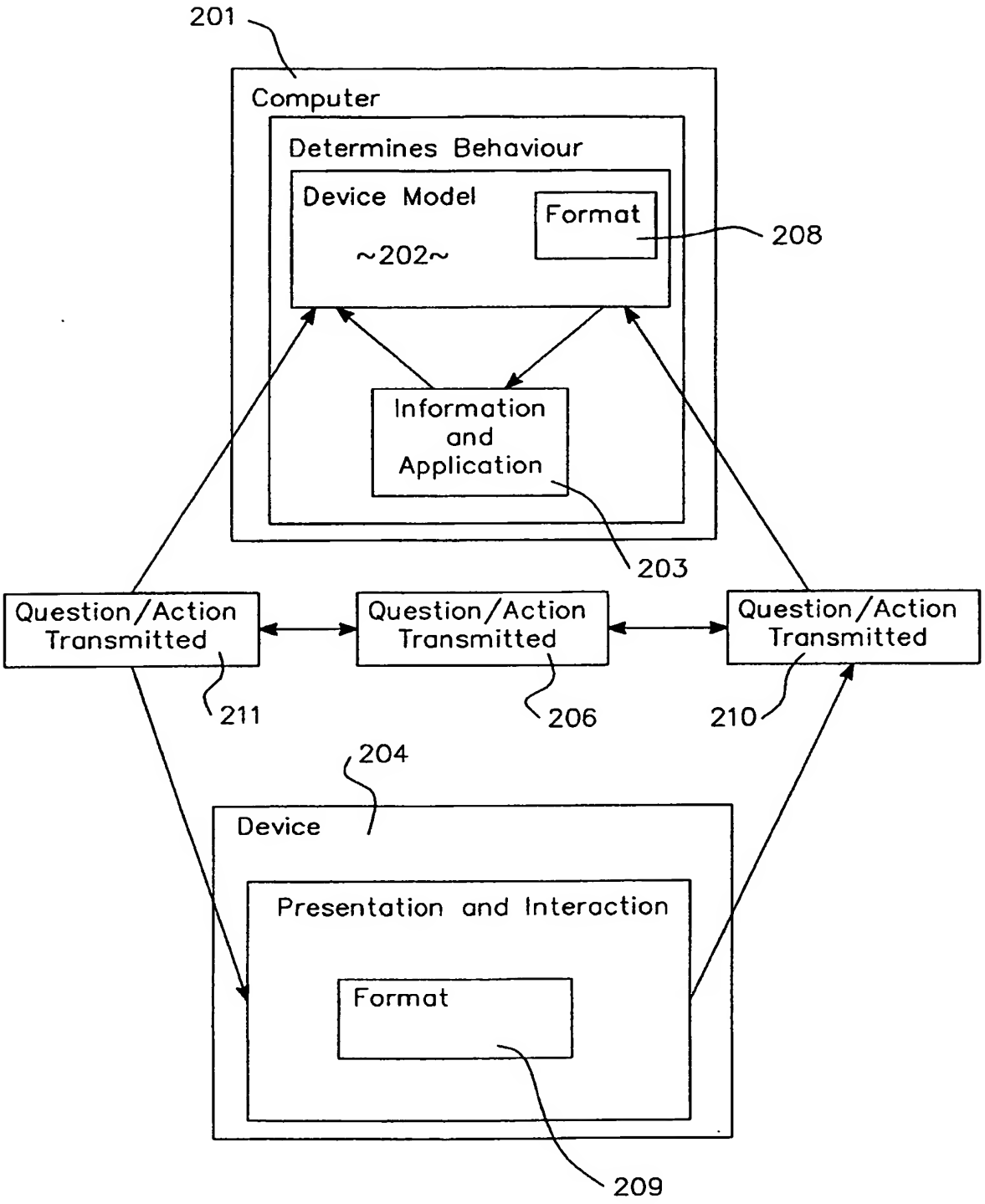

201
Computer
Determines Behaviour
Device Model
~202~
Format
208
Information and Application
203
Question/Action Transmitted
Question/Action Transmitted
Question/Action Transmitted
211
206
210
204
Device
Presentation and Interaction
Format
209

Fig. 11

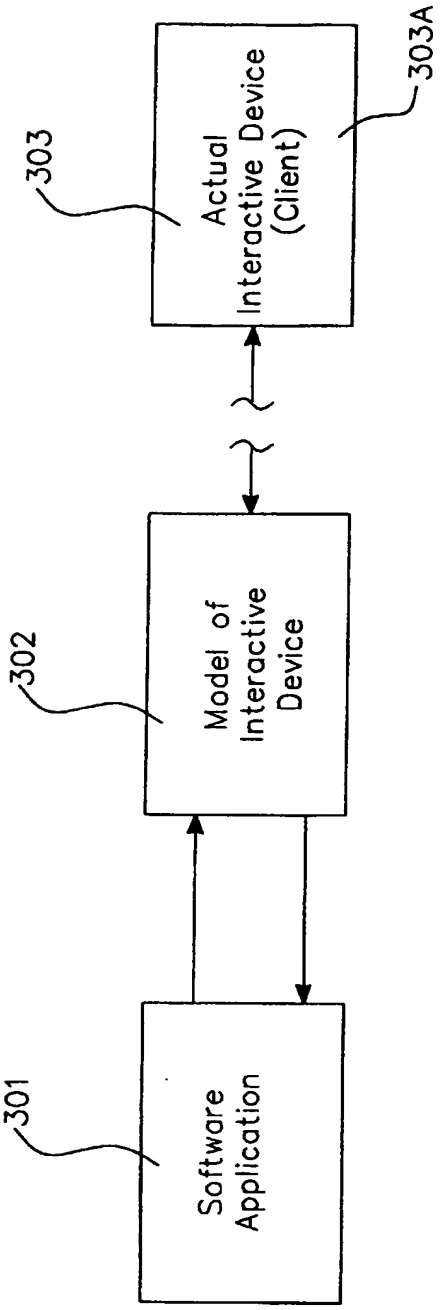

303A
303
Actual Interactive Device (Client)
302
Model of Interactive Device
301
Software Application

Fig. 12

INTERNATIONAL SEARCH REPORT

International application No.
PCT/AU00/00983

A. CLASSIFICATION OF SUBJECT MATTER

Int. Cl. [7]: G06F 15/16

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
IPC G06F 15/16, 15/163, 15/167, 15/17, 15/173, 15/177

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched
AU : IPC AS ABOVE

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)
WPAT, USPTO Web Patent Database, " library, remote, application, program, network, distributed, client, server etc."

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	US, A, 5768510 (GISH) 16 June 1998 Column 17 line 11 to column 19 line 63 and the claims in particular.	1-50,102-106
X	WO, A, 99/40519 (MERRILL LYNCH & CO. INC.) 12 August 1999 Page 8 lines 16 to 25 and page 9 line 28 to page 10 line 21 in particular.	1-50,102-106
X	US, A, 5379374 (ISHIZAKI et al.) 3 January 1995 See citation's claims in particular.	1-50,102-106
X Y	EP, A, 592 080 (INTERNATIONAL BUSINESS MACHINES CORPORATION) 13 April 1994 Entire document. In combination with the above citations.	1-3 4-50,102-106

[X] Further documents are listed in the continuation of Box C [X] See patent family annex

* Special categories of cited documents:

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier application or patent but published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art

"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
23 October 2000	- 7 NOV 2000
Name and mailing address of the ISA/AU AUSTRALIAN PATENT OFFICE PO BOX 200, WODEN ACT 2606, AUSTRALIA E-mail address: pct@ipaustralia.gov.au Facsimile No. (02) 6285 3929	Authorized officer P. THONG Telephone No : (02) 6283 2128

INTERNATIONAL SEARCH REPORT	International application No. PCT/AU00/00983

C (Continuation). DOCUMENTS CONSIDERED TO BE RELEVANT		
Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	US, A, 5586117 (EDEM et al.) 17 December 1996 Column 4 lines 34 to 44, column 12 line 26 to column 14 line 37 in particular.	51-97
X	US, A, 5586937 (MENASHE) 24 December 1996 Column 5 lines 60 to 67 and column 6 lines 19 to 49 in particular.	9,34
X Y	US, A, 5801689 (HUNTSMAN) 1 September 1998 Column 8 line 66 to column 10 line 26 in particular In combination with US 5768510 for example.	9,34,51-102,105,106 1-8,10-33,35-50,103,104
X	US, A, 5580177 (GASE et al.) 3 December 1996 Whole document.	103,104

INTERNATIONAL SEARCH REPORT

International application No.
PCT/AU00/00983

Box I **Observations where certain claims were found unsearchable (Continuation of item 2 of first sheet)**

This international search report has not been established in respect of certain claims under Article 17(2)(a) for the following reasons:

1. ☐ Claims Nos :

because they relate to subject matter not required to be searched by this Authority, namely:

2. ☐ Claims Nos :

because they relate to parts of the international application that do not comply with the prescribed requirements to such an extent that no meaningful international search can be carried out, specifically:

3. ☐ Claims Nos :

Because they are dependent claims and are not drafted in accordance with the second and third sentences of Rule 6.4(a)

Box II **Observations where unity of invention is lacking (Continuation of item 3 of first sheet)**

This International Searching Authority found multiple inventions in this international application, as follows:

(1) Claims 1,3,18,25,26,28,43,50,103,104 and dependent claims: Arrangement involving local and/or remote library.
(2) Claims 9,34 and dependent claims: Arrangement involving bi-directional communication.
(3) Claims 51 and dependent claims. Arrangement with re-configuration capability.
(4) Claims 98, 100 and dependent claims. Arrangement for user interface.

1. ☐ As all required additional search fees were timely paid by the applicant, this international search report covers all searchable claims
2. ☒ As all searchable claims could be searched without effort justifying an additional fee, this Authority did not invite payment of any additional fee.
3. ☐ As only some of the required additional search fees were timely paid by the applicant, this international search report covers only those claims for which fees were paid, specifically claims Nos.:

4. ☐ No required additional search fees were timely paid by the applicant. Consequently, this international search report is restricted to the invention first mentioned in the claims; it is covered by claims Nos.:

Remark on Protest

☐ The additional search fees were accompanied by the applicant's protest.

☐ No protest accompanied the payment of additional search fees.

INTERNATIONAL SEARCH REPORT
Information on patent family members

International application No.
PCT/AU00/00983

This Annex lists the known "A" publication level patent family members relating to the patent documents cited in the above-mentioned international search report. The Australian Patent Office is in no way liable for these particulars which are merely given for the purpose of information.

Patent Document Cited in Search Report		Patent Family Member					
US	5768510	EP	822488	JP	10312284		
WO	99/40519	AU	25671/99	US	6047324		
US	5379374	JP	4186456				
EP	592080	JP	6295290	US	5544316.		
US	5586937	AU	63184/94	CA	2123857	DE	69421315
		ES	2141199 .				

END OF ANNEX